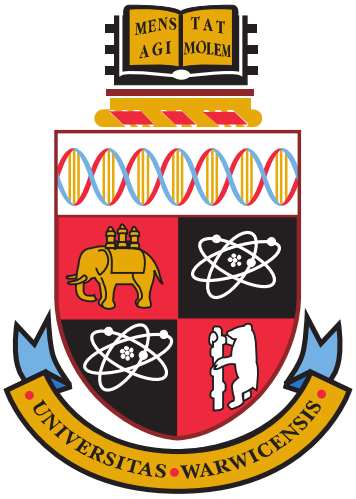


A study of the alignment of polymeric molecules and vesicles in fluid flows for linear dichroism experiments

by

Don Praveen Amarasinghe

Supervisors: **Dr. Nikola P. Chmel, Prof. Alison Rodger, Dr. Vasily Kantsler and Prof. Ravi Jagadeeshan**

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Molecular Organisation and Assembly in Cells (MOAC) Doctoral Training Centre, University of Warwick

January 2018

Contents

Contents i

List of Tables vi

List of Figures ix

List of Abbreviations xx

Acknowledgements xxiii

Declaration xxv

Abstract xxvi

1 Introduction 1

1.1 Absorbance and linear dichroism 2

1.2 LD experiments with polymers in solution 9

1.3 Simulating polymers in dilute solutions 10

1.4 Studying the structure of membrane proteins 16

1.5 Aligning vesicles in fluid flow 21

1.6 Microfluidic experiments with vesicles 28

1.7 Thesis outline 32

2 Using FENE-Fraenkel springs in polymer simulations 33

2.1 Spring Force Law Formulation 34

2.1.1 FENE springs 34

2.1.2 Fraenkel springs 35

2.1.3 FENE-Fraenkel stiff spring 37

2.2 Non-dimensionalised forms of the spring force laws 39
2.2.1 FENE force law 39
2.2.2 Fraenkel force law 40
2.2.3 FENE-Fraenkel force law 40
2.3 Regenerating the FENE and Fraenkel force laws 41
2.3.1 Obtaining the FENE law from FENE-Fr. law 41
2.3.2 Obtaining the Fraenkel law from FENE-Fr. law 41
2.4 Zero-shear rate equilibrium properties 42
2.4.1 Calculating equilibrium properties 42
2.4.2 Potential function ϕ for a dumbbell 44
2.4.3 Equilibrium properties of the length of a FENE-Fraenkel spring . 45
2.5 Excluded Volume Force Law 54
2.6 Hydrodynamic interactions 55
2.7 Numerical method for solving the bead-spring configuration 56
2.7.1 Bead-Spring Configuration Equation 56
2.7.2 Discretisation — Step 1 — First guess 59
2.7.3 Discretisation — Step 2 — Updated guess 60
2.7.4 Discretisation — Step 3 — Recasting the updated guess 60
2.7.5 Discretisation — Step 4 — Generating a cubic 62
2.8 Summary 63

3 Simulating polymers in flow with FENE-Fraenkel springs 64
3.1 Code alterations to incorporate the FENE-Fraenkel spring 64
3.2 Verification of simulation against analytically derived properties 66
3.2.1 Zero-shear conditions 66
3.2.2 Non-zero shear conditions 69
3.3 Comparison against work by Hsieh et al. [48] 72
3.3.1 Conversion between parameter values 72
3.3.2 Choice of H^* and its influence on simulated polymer stresses . . . 75
3.3.3 Choice of δ^* and its influence on simulated polymer stresses . . . 79
3.3.4 Choice of Δt^* and its influence on simulated polymer stresses . . 83
3.3.5 Summary of comparisons with data from Hsieh et al. [48] 88

3.4 Orientation behaviour of a FENE-Fraenkel dumbbell in linear shear flows with and without hydrodynamic interactions 89
3.5 Summary and future work 100

4 Development of planar microfluidic devices for linear dichroism studies 102
4.1 LD experiments using Couette Flow 103
4.2 Absorbance of PDMS 104
4.3 Preparation of planar microfluidic channels 106
4.3.1 Making the silicon wafer cast 106
4.3.2 Manufacture of the PDMS microfluidic channel from the cast . . 110
4.4 Preliminary experiments with microfluidic devices 111
4.4.1 Results with wide channel 113
4.4.2 Results with zig-zag channel 116
4.4.3 Analysis of results 119
4.5 Development of an extensional flow device 120
4.6 Measuring and modifying the path length of microfluidic devices 124
4.6.1 Using light inteferometry to measure channel depth of devices . . 124
4.6.2 Increasing path length and measuring channel depths using light interferometry 126
4.6.3 Using potassium chromate absorbance to measure channel depth . 128
4.7 Experimental results with deeper microfludic devices 130
4.7.1 Couette shear flow 130
4.7.2 Poiseuille (pressure-driven) flow in the deep wide channel 132
4.7.3 Extensional flow in the deep cross-slot 134
4.7.4 Comparison of orientation performance 136
4.8 Summary and future work 138

5 Linear dichroism experiments with vesicles in flow 139
5.1 Introduction — Lipids, vesicles and DPH 140
5.2 Initial protocol for vesicle preparation 143
5.3 Experiments with Couette flow and the wide channel microfluidic device . 146
5.4 Experiments with a shallow cross-slot microfluidic device 149
5.5 Vesicle preparation and method development 155

5.6 Tests with pressure-driven and extensional flows with the deep wide channel and cross-slot devices . . . 156
5.6.1 Protocol for vesicle preparation . . . 156
5.6.2 Pressure-driven flows through the deep wide channel device . . . 157
5.6.3 Extensional flows through the deep cross-slot device . . . 162
5.6.4 Discussion . . . 166
5.7 Summary and future work . . . 168

6 Conclusions and future work **169**
6.1 Implementation of the FENE-Fraenkel force law into bead-spring chain simulations . . . 169
6.2 Development of an extensional flow microfluidic device for use in LD experiments . . . 171

Bibliography **174**

A Derivation of Equation (2.4.11) **190**

B Fortran code documentation **193**
B.1 sensemble.f90 . . . 193
B.2 modules.f90 . . . 196
B.3 utils.f90 . . . 196
B.4 gsipc.f90 . . . 198
B.5 properties.f90 . . . 201

C Alterations to the code to incorporate the FENE-Fraenkel spring **203**
C.1 sensemble.f90 . . . 204
C.2 modules.f90 . . . 205
C.3 utils.f90 . . . 207
C.4 gsipc.f90 . . . 212
C.5 properties.f90 . . . 227

D Fortran code for simulations **228**
D.1 sensemble.f90 . . . 228
D.2 modules.f90 . . . 245

D.3 utils.f90 . . . 252
D.4 gsipc.f90 . . . 270
D.5 properties.f90 . . . 299

E Vesicle preparation and method development 305
E.1 Vesicle size during manufacture . . . 305
E.1.1 Slow cycles . . . 307
E.1.2 Fast cycles . . . 309
E.2 Using purified lipids . . . 310
E.2.1 Using DOPC . . . 310
E.2.2 Using POPC . . . 312

F Supplementary figures from experiments with vesicles in Chapter 5 314
F.1 Additional figures referenced in Section 5.3 . . . 314
F.2 Data for vesicles without DPH in pressure-driven flow for Subsection 5.6.2 318
F.3 Data for vesicles without DPH in extensional flow for Subsection 5.6.3 . . 320

List of Tables

3.1 Parameters used for simulation testing to see if analytical results for the FENE-Fraenkel are obtained. 66

3.2 Mean values and standard errors for $\langle R^2 \rangle$ with zero shear rate. These have been averaged over all timepoints. 69

3.3 Mean values and standard errors for $\langle R^2 \rangle$ under different linear shear rates and using a time step of 1 time unit. These have been averaged over the timepoints of the last 2000 time units of the 10000 time unit length simulation. 70

3.4 Values of various parameters under the non-dimensionalisation paradigm of this work converted from the parameters used in [48] to analyse the effect of the choice of spring constant parameter on simulation results. . . 76

3.5 Averaged values for the mean and standard error of σ_{11} between timepoints 1.8 and 2.0 (the "plateau") in simulations using parameters described in Table 3.4 and in [48, Figure 3]. 78

3.6 Values of various parameters under the non-dimensionalisation paradigm of this work converted from the parameters used in [48] to analyse the effect of the choice of spring extensibility parameter on simulation results. 80

3.7 Averaged values for the mean and standard error of σ_{11} between timepoints 2.4 and 2.5 (the "plateau") in simulations using parameters described in Table 3.6 and in [48, Figure 4]. 80

3.8 Values of various parameters under the non-dimensionalisation paradigm of this work converted from the parameters used in [48] to analyse the effect of the time-step size on simulation results. 83

3.9 Values of Δt^* tested to ascertain the effect of time-step size on our simulations, with values of the average mean and standard error for σ_{11} between 1.97 and 2.03 time units (as scaled according to the simulation results of Hsieh et al. [48]). 86

3.10 Parameters used for simulations to assess the alignment of FENE-Fraenkel dumbbells in shear flow. 89

3.11 Mean values and standard errors for the orientation parameter S (to 3 decimal places) under different linear shear rates, different time-step sizes and with or without hydrodynamic interactions present. These have been averaged over the timepoints of the last 20000 time units of the 100000 time unit length simulation. 94

4.1 Relationship between duration of soft bake stages and the thickness of the SU-8 2100 photoresist layer desired. This table is reproduced with data from Table 2 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem. 108

4.2 Relationship between UV light exposure energy doses and the thickness of the SU-8 2100 photoresist layer desired. This table is reproduced with data from Table 3 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem. 108

4.3 Relationship between duration of post exposure bake stages and the thickness of the SU-8 2100 photoresist layer desired. This table is reproduced with data from Table 5 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem. 109

5.1 Minimum flow rates required for flow in the cross-slot design to overcome Brownian rotational forces, on the assumption of treating vesicles as spheres. 151

5.2 Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 25 samples. 168

B.1 Input variables in lines 77 to 103 of sensemble.f90. 194

E.1 Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 25 samples at each stage. 306
E.2 Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 14 samples after each slow freeze-thaw cycle. 308
E.3 Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 15 samples after each slow freeze-thaw cycle. 309
E.4 Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 25 samples after extrusion. 312

List of Figures

1.1 An illustration of unpolarised and polarised light. A polariser is used to allow light of a particular polarisation to pass through (in this case, vertically polarised light); all other polarisations are blocked. 4

1.2 Dimensions of the Couette flow cell used for LD experiments. 7

1.3 Schematic of a Couette flow cell for LD experiments. 7

1.4 Example of a bead-spring chain with seven beads. In the Rouse model, these springs are Hookean (the tension force is proportional to their length) and freely-jointed to the beads (meaning that there is no restriction on the angle between two springs attached to the same bead). 11

1.5 Graphs of $\cot 2\chi$ and $-\Delta A_{45}/A$ against G, where χ is the angle between the electronic transition of interest and the flow direction (corresponding to α in Equation (1.1.4)), A is the isotropic absorbance, ΔA_{45} is the change in absorbance observed between the flow direction and 45° to the flow direction and G is the shear rate. These experimental data were obtained by studying DNA in flow. The lines plotted are using linear least-squres regression. Theory from Rouse-Zimm models suggests a linear relationship between G and observable variables $\cot\chi$ and $-\Delta A_{45}/A$. This linear relationship is seen for low shear rates, but degenerates for values of G above $20\,\mathrm{s}^{-1}$. These figures are reproduced from Figures 1 and 2 of [164]. 14

1.6 Diagram defining the axes referred to in the expression for the orientation parameter S (Equation (1.3.4)). 15

1.7 Illustration of the basic components of phospholipids and a schematic cross-section of a phospholipid bilayer with membrane proteins. 17

1.8 Illustration of the hierarchy of protein structure. All proteins are comprised of at least one chain of amino acid units (a *polypeptide chain*); most proteins are made up of multiple chains. The sequence of these amino acids in a chain is known as the *primary structure*. A polypeptide chain is usually folded up into a three-dimensional form, containing substructures known as *α-helices* and *β-sheets* held together by hydrogen-bonding; these make up the protein's *secondary structure*. Further folding and arrangements of the secondary structure gives rise to the *tertiary structure*. The collection of tertiary structures that make up a protein are known as the *quartenary structure*. A more detailed treatment of protein structure can be found in [75]. 18

1.9 Vesicle types based on diameter and lamellarity. This diagram is reproduced with permission from the Figure 2 of [55]. 21

1.10 An illustration of the three types of behaviour that vesicles undergo in planar fluid-flow. The three lines corresponds with (from top to bottom) tank-treading (TT), trembling (TR) and tumbling (TB). These diagrams are reproduced with permission from the abstract figure of [2]. 22

1.11 A diagram illustrating the rotational (left) and extensional/elongational (right) components of a linear shear flow (middle). Reproduced from Figure 2 of [2]. 24

1.12 Phase diagram illustrating the regions of (S, Λ)-parameter space that correspond to the different vesicle behaviours. Tank-treading takes place in the blue and green regions, trembling in the area between solid red and broken blue lines and tumbling in the orange and grey areas. The diagram is reproduced from Figure 9 of [71]. 25

1.13 The elongation of a vesicle containing chromophores of interest in its membrane results in a net orientation of chromophores. In this example, the spherical vesicle has chromophores (in red) distributed evenly throughout its membrane, and the net orientation of the chromophores is zero. Elongation results in more of these chromophores occupying the parts of the membrane along the long axis of the vesicle, so the net orientation of the chromophores is close to vertical. 26

1.14 An illustration of the tilt of a vesicle away from a stationary boundary wall while in fluid shear flow. The vesicle is experiencing a lift force F_{lift}, whose magnitude is dependent upon the shear rate $\dot{\gamma}$. This diagram is reproduced with permission from the abstract figure of [2]. 27

1.15 This diagram illustrates the cross-slot device used by Tanyeri et al. [147]. (a) A hydrodynamic trap is created by a planar extensional flow field at the junction of two perpendicular microchannels. (b) The velocity field (top) and the potential well (bottom) for a particle in the flow field at the microchannel junction. This diagram is reproduced with permission from Figure 1 of [147]. 29

1.16 The four-roll mill device used by Taylor [150]. Four rotating mills controlled by driving pulleys cause the fluid drop in the centre to change shape. This diagram is reproduced with permission from Figure 1 of [150]. 30

1.17 (a) A macroscale summary of the four-roll mill. (b) The microfluidic four-roll mill device used by Lee et al. [72]. All flow types can be generated by varying the angular velocity ratio ω_1/ω_2 or flow rate ratio Q_1/Q_2. These diagrams are reproduced with permission from Figure 1 of [72]. 30

2.1 An illustration of a spring vector of a spring in a bead-spring chain. . . . 34

2.2 An illustration of the relationship between the length and the spring tension of a FENE spring. The spring length can vary between zero inclusive and $Q_{\max}$ exclusive. The tension force increases in magnitude significantly as the spring length gets close to $Q_{\max}$. 35

2.3 An illustration of the relationship between the length and the spring tension of a Fraenkel spring. The spring tension/compression force is linearly proportional to the extension/compression from the natural spring length Q_0. 36

2.4 An illustration of the relationship between the length and the spring tension of a FENE-Fraenkel spring. The tension/compression force is zero when the spring is at its natural length Q_0, and increases in magnitude with stretching/compression from this length, approaching infinity when the spring length gets close to $Q_0 - \delta$ or $Q_0 + \delta$. 38

3.1 Plots of the squared spring length $\langle R^2 \rangle$ against time for different time-step values and different combinations of natural length Q_0 and extensibility δ under zero shear conditions and a line indicating the analytically derived value. Standard error bars have been ommitted for clarity. 68

3.2 Plots of the squared spring length $\langle R^2 \rangle$ against time for different time-step values and different shear rates and a line indicating the analytically derived value under zero shear conditions, using a FENE-Fraenkel spring with natural length $Q_0 = 200$ and extensibility $\delta = 10$. Standard error bars have been ommitted for clarity. Means and standard errors for the last 2000 time units of each simulation are given in Table 3.3. 71

3.3 Plots of σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, corresponding to different parameter combinations described in Table 3.4. Time scales of the simulations have been rescaled to match that of the data given in Hsieh et al. [48], each corresponding to a different value of spring constant (spring stiffness). The "Hsieh data" plot corresponds to data obtained from Figure 3 in Hsieh et al. [48] 77

3.4 Histogram of the deviations from the average value of σ_{11} along the plateau of plots in Figure 3.3. 79

3.5 Plots of σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow with different spring extensibility values, as given in Table 3.6. Time scales of the simulations have been rescaled to match that of the data given in Hsieh et al. [48]. The "Hsieh data" plot corresponds to data obtained from Figure 4 in Hsieh et al. [48] . 81

3.6 Histogram of the deviations from the average value of σ_{11} along the plateau of plots in Figure 3.5. 81

3.7 Magnification of the plots of Figure 3.5 in the plateau region between 2.4 and 2.5 time units. The lower plot features simulation data sampled at fewer time points, but includes the standard error bars at those points. These are based on a sample size of 10000 independent trajectories. Hsieh data obtained from [48, Figure 4 Inset] using Plot Digitizer. 82

3.8 Plots of σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, corresponding to different time-step sizes, as described in Table 3.8. The "Hsieh data" plot corresponds to data obtained from [48, Figure 7]. Time scales of the simulations have been rescaled to match that of the data given in [48]. 84

3.9 Magnification of the plots of Figure 3.5 in the plateau region between 1.97 and 2.03 time units, including standard error bars at each time point (i.e. Equation (3.2.1) is applied but with data with respect to a single time point), for our simulations based on a sample size of 1000 independent trajectories. Hsieh data obtained from [48, Figure 7 Inset] using Plot Digitizer. 84

3.10 Plots of the σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, restricted to the plateau region between 1.97 and 2.03 time units (as scaled according to the simulation results of Hsieh et al. [48]) for a range of values of Δt^* as detailled in Table 3.9. The simulations were based on a sample size of 1000 independent trajectories. 86

3.11 Plots of the σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, as in Figure 3.10, for a selection of values of Δt^*. Standard error bars at a selection of time points are included (Equation (3.2.1) is applied but with data with respect to a single time point). 87

3.12 Plots of the orientation parameter S against time for different time-step values and different linear shear rates $\dot{g}$ without hydrodynamic interactions, using a FENE-Fraenkel spring with natural length $Q_0 = 100$ and extensibility $\delta = 1$. Standard error bars have been ommitted for clarity. . . 95

3.13 Plots of the orientation parameter S against time for different time-step values and different linear shear rates $\dot{g}$ with hydrodynamic interactions ($h^* = 28.211$), using a FENE-Fraenkel spring with natural length $Q_0 = 100$ and extensibility $\delta = 1$. Standard error bars have been ommitted for clarity. 96

3.14 Plots of the orientation parameter S against time with a shear rate value of $\dot{g} = 1\times10^{-4}$ for different time-step values with and without hydrodynamic interactions, using a FENE-Fraenkel spring with natural length $Q_0 = 100$ and extensibility $\delta = 1$. Standard error bars have been ommitted for clarity. 97

3.15 Plots of the orientation parameter S against time for all shear rate values tested (labelled SR in the diagram) with and without hydrodynamic interactions, using a FENE-Fraenkel spring with natural length $Q_0 = 100$, extensibility $\delta = 1$ and time-step size $\Delta t = 0.5$. Standard error bars have been ommitted for clarity. 98

3.16 Plots of the orientation parameter, S, against time for different particle sizes and shear rates at 22°, as calculated from the model developed by McLachlan et al. [89]. a and b are the particle semi-diameters measured parallel and perpendicular, respectively, to the axis of revolution, k is the shear rate and λ is the particle material time constant (see [14]). Subfigure (a) concerns shorter particles ($b = 4\,\text{nm}$) or low shear rates. Subfigure (b) focusses on high shear rates for M13 bacteriophage particles ($\lambda = 4.37\,\text{ms}$). The graphs are reproduced from Figure 2 of [89]. 99

4.1 Schematic of the Couette flow cell used for LD experiments. 103

4.2 Schematic of a Couette flow cell for LD experiments. 104

4.3 Absorbance spectra of polydimethylsiloxane (PDMS) samples with different thicknesses. 105

4.4 Relationship between final wafer rotation speed and the thickness of the SU-8 2100 photoresist layer desired. This figure is reproduced from Figure 1 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem. The blue line corresponds to the photoresist used in this project. 107

4.5 A photo of a Karl Suss MJB3 Mask Aligner, used for the exposure of silcon wafers with a photoresist layer to UV light. The mask is placed, photoresist layer on top, on the silver circular holder on the red plate. This is then moved into place under a glass in the exposure zone (located underneath the microscope). A vacuum holds the silicon wafer in place. The X-ray film with the design printed as a negative is placed on the glass just underneath the microscope — the "shiny" film side should be placed facing up. 109
4.6 An example of a finished silicon wafer cast with channel designs in photoresist in the centre. Scotch® tape is attached to the wafer to form a well for the liquid PDMS to set. 110
4.7 "Wide Channel" 111
4.8 "Zig-zag channel" 111
4.9 Illustration of the use of an IR spectroscopy cell holder for the flow cell. . 112
4.10 Linear dichroism spectra of DNA solution passed through the wide channel microfluidic device at different flow rates. 113
4.11 Absorbance of DNA solution passed through the wide channel microfluidic device. 114
4.12 Reduced linear dichroism spectra of DNA solution passed through the wide channel microfluidic device at different flow rates. 114
4.13 Estimated values of the orientation parameter (calculated using the reduced LD signal measured at 260 nm) against flow rate using the wide channel microfluidic device. 115
4.14 Linear dichroism spectra of DNA solution passed through the zig-zag channel microfluidic device at different flow rates. 116
4.15 Absorbance of DNA solution passed through the zig-zag channel microfluidic device. 117
4.16 Reduced linear dichroism spectra of DNA solution passed through the zig-zag channel microfluidic device at different flow rates. 117
4.17 Estimated values of the orientation parameter (calculated using the reduced LD signal measured at 260 nm) against flow rate using the zig-zag channel microfluidic device. 118

4.18 Basic schematic of a cross-slot with extensional flow in the centre. The left diagram shows the flow directions. The right diagram illustrates the behaviour of polymer molecules in extensional flow. 120
4.19 The design for the extensional-flow microfluidic device used in experiments in this thesis. The left diagram is the complete design of the channels embedded in PDMS. The right diagram is a zoomed-in view of the central intersection, with arrows indicating the flow directions. 121
4.20 Details of particular lengths and radii of curvature for the Haward cross-slot design used in the experiments reported in this thesis. 122
4.21 Surface profiler depth map of the orignal wide channel used for preliminary tests. 125
4.22 Dimensions of the revised wide channel design. 126
4.23 Surface profiler depth map of the deeper wide channel design on the silicon wafer, as used for later experiments. The tilt that can be observed here is due to the configuration of the platform on which the wafer was placed, so as to allow the inteferometer to capture the light interference waveforms precisely. 127
4.24 Surface profiler depth map of the deeper cross-slot used for final tests. . . 127
4.25 Absorbance spectrum of 2 mM potassium chromate solution passes through the cross-slot device. 129
4.26 Absorbance spectra of 2 mM potassium chromate solution through the wide channel microfluidic device as it is pumped at different flow rates. . 129
4.27 Derived path length of the wide channel microfluidic device at different flow rates. 130
4.28 Linear dichroism, absorbance and reduced LD spectra of DNA solution in Couette flow at different rotation speeds. The measurements were taken with DNA solution prepared for use in the deep cross-slot microfluidic device. The results for the solution prepared for use in the deep wide channel device gave similar value for the reduced LD. 131
4.29 Linear dichroism & reduced LD spectra of DNA solution in pressure-driven flow through the deep wide channel microfluidic device and a plot of reduced LD at 260 nm against flow rate. 133

4.30 Linear dichroism, absorbance and reduced LD spectra of DNA solution in extensional flow through the deep cross-slot microfluidic device. 135
4.31 Linear dichroism & reduced LD spectra of DNA solution in extensional flow through the deep wide channel microfluidic device and a plot of reduced LD at 260 nm against flow rate. 137

5.1 Illustration of the basic components of phospholipids and a cross-section of a phospholipid bilayer. 140
5.2 Distribution of fatty acids present in soy bean PC sourced from Sigma Aldrich® P3644. The numerical X:Y labels refer to the number of carbons in the fatty acid chain (X) and the number of unsaturated carbon double bonds present in the chain (Y). 141
5.3 Absorbance spectrum of 1,6-diphenylhexa-1,3,5-triene (DPH) in methanol ($0.02\,\mathrm{mg\,ml^{-1}}$), with a diagram of a DPH molecule and an inset of the peaks between 250 nm and 280 nm. Peaks corresponding to long- and short-axis transitions are marked on the plot. 142
5.4 Diagram and photos of the Avanti Polar Lipids Mini-Extruder Kit used to extrude lipid vesicle solutions. 145
5.5 Absorbance and LD spectra of soy bean PC vesicles containing DPH (0%, 2%, 4% and 10% by mass) in Couette shear flow and pressure-driven flow through the wide channel microfluidic device. The LD spectra of soy bean PC vesicles with 4% and 10% DPH by mass in pressure-driven flow are in the Appendices in Figure F.1 146
5.6 Vesicle size distribution data obtained using DLS for soy bean PC vesicles containing 0% and 2% DPH by mass before use in flow experiments. . . . 148
5.7 Structures of three PC lipids with chains typical to those found in soy bean PC. 152
5.8 Absorbance and LD spectra of soy bean PC vesicles without DPH and with DPH (0.4% by mass) in extensional flow in the cross-slot microfluidic device, and vesicle size distribution data obtained using DLS for soy bean PC vesicles containing no DPH before use in measurements using the cross-slot device. 153

5.9 Linear dichroism and absorbance spectra of soy bean PC vesicles with 0.4% DPH by mass, in pressure-driven flow through the deep wide channel microfluidic device at different flow rates. 159
5.10 Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep wide channel microfluidic device. 160
5.11 Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep wide channel microfluidic device. 161
5.12 Linear dichroism and absorbance spectra of soy bean PC vesicles with 0.4% DPH by mass, in extensional flow through the deep cross-slot microfluidic device at different flow rates. 163
5.13 Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep cross-slot microfluidic device. 164
5.14 Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep cross-slot microfluidic device. 165

E.1 Vesicle diameter distribution data (measured using intensity measurements obtained with DLS) for soy bean PC vesicles after individual stages of the manufacture protocol. 306
E.2 Vesicle diameter distribution data (measured using intensity measurements obtained with DLS) for soy bean PC vesicles after individual stages of the manufacture protocol. 308
E.3 Vesicle diameter distribution data (measured using intensity measurements obtained with DLS) for soy bean PC vesicles after individual stages of the manufacture protocol. 309
E.4 Vesicle size distribution data obtained using DLS for DOPC vesicles with 1.0% DPH by mass after one slow and three fast freeze-thaw cycles and extrusion through membrane with 100 nm diameter pores. Error bars indicate the standard deviation across 21 samples. 311

E.5 Absorbance spectra two samples of DOPC vesicles with 1% DPH by mass in Couette shear flow. 311
E.6 Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles and POPC vesicles with 0.4% DPH by mass. Error bars indicate the standard deviation across 25 samples. . . 312
E.7 Linear dichroism and absorbance spectra from Couette flow for soy bean PC and POPC vesicles with 0.4% DPH by mass, prepared after one slow and three fast freeze-thaw cycles and extrusion through membrane with 100 nm diameter pores. 313

F.1 LD spectra of soy bean PC vesicles containing DPH (4% and 10% by mass) in pressure-driven flow in the wide channel microfluidic device. . . 315
F.2 Vesicle size distribution data obtained using DLS for soy bean PC vesicles containing 4% DPH by mass before use in flow experiments. 316
F.3 Vesicle size distribution data obtained using DLS for soy bean PC vesicles containing 10% DPH by mass before use in flow experiments. 317
F.4 Linear dichroism and absorbance spectra of soy bean PC vesicles in pressure-driven flow through the deep wide channel microfluidic device at different flow rates. 318
F.5 Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles, before and after use in the deep wide channel microfluidic device. 319
F.6 Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles, before and after use in the deep wide channel microfluidic device. 319
F.7 Linear dichroism and absorbance spectra of soy bean PC vesicles in extensional flow through the deep cross-slot microfluidic device at different flow rates. 320
F.8 Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles, before and after use in the deep cross-slot microfluidic device. 321
F.9 Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles, before and after use in the deep cross-slot microfluidic device. 321

List of Abbreviations

CD	Circular dichroism
DLS	Dynamic Light Scattering
DNA	Deoxyribonucleic Acid
DPH	1,6-Diphenylhexa-1,3,5-triene
FENE	Finitely extensible nonlinear elastic
FENE-Fr.	FENE-Fraenkel
GUV	Giant unilamellar vesicle
HI	Hydrodynamic interactions
HPLC	High-performance liquid chromatography
IR	Infra-red
LD	Linear dichroism
LHS	Left-hand side
LUV	Large-sized unilamellar vesicle
MLV	Multilamellar vesicle
MUV	Medium-sized unilamellar vesicle
NMR	Nuclear magnetic resonance
ODE	Ordinary differential equation
PC	Phosphatidylcholine

PDE	Partial differential equation
PDMS	Polydimethylsiloxane
RHS	Right-hand side
rpm	Rotations (or revolutions) per minute
SUV	Small-sized unilamellar vesicle
UV	Ultra-violet

This thesis is dedicated to the memory of my grandfather, Nelson Perera Samaranayake, who passed away before final publication.

මම ඔහුට නිවන්සුව පතමි!

Acknowledgements

First, and foremost, my thanks go to my family — Amma, Thatha, Malee, Archi and Seya. Their support, love and occasional (read "persistent") nagging has always kept me going. Thanks must also go to everyone involved in the MOAC programme and its sister doctoral training centres (Marie Curie CAS:IDP, MAS, SysBio and IBR) over the last 5 years. Special mention must go to Naomi Grew in administration; a friendly, welcoming face in Senate House! I am grateful to the Engineering and Physical Sciences Research Council for funding my position in MOAC.

Thanks are due to Adam Hall and Dave McCormick of MASDOC, a CDT in the mathematical sciences at Warwick, for their guidance in helping me typeset this dissertation with LaTeX. I also appreciate the friendship and support that I have received from close friends in the Mathematics Institute and Department of Statistics throughout my time at Warwick. My supervisees from many years of teaching in the Maths Institute also deserve a mention, some of whom have gone on to become dear friends.

I would also like to thank my many friends both at the University and outside it for their unwavering support and their friendship — I can only apologise to them all for not having enough space to include them here! Thanks go to the Music Centre at the University for providing invaluable time away from degree work. Paul McGrath, Lucy Griffiths, Lucy Young, the vocal tutors, the Chamber Choir, the Chorus, Opera Warwick and the many other instrumental and vocal ensembles that I have either been a part of or worked with. It has been a joy and privilege to work and make music with some of the most talented and gifted musicians in the country throughout the 12 years that I have been at Warwick.

After seven years at Warwick, and only having tried one sport for only a term in first year, I took the opportunity at the start of my PhD to join (and commit properly to) another

sports club. I am forever grateful that I picked the University of Warwick Trampoline Club and to all the friends that I have made there. At a time when I was worried that I really was overstaying my welcome at Warwick, I was very happy to discover a new set of friends and, unexpectedly, develop an unusual skill! Particular mention should go to all the coaches at the club — again, most have become very close friends.

As part of my PhD studies, I was fortunate enough to be asked to fly to Melbourne in Australia, and work with Prof. Ravi Jagadeeshan at Monash University. Whilst there have been times where we haven't seen eye-to-eye (and I still don't see the appeal of Fortran!), I am very grateful to him for giving me the opportunity to work with him and for providing me with the initial bead-spring code to modify. The computational outputs of this research are due to the resources provided by Melbourne Bioinformatics at the University of Melbourne under grant number VR0010. I also want to thank Chandi, Chamath and Emma in Ravi's group for their support with my research. Special mention also goes to MonUCS, MOVE Monash and various Warwick students who were present in Monash during my stay there, for making sure that I was not lonely and bogged down in Fortran during my time abroad!

The penultimate group of people I want to thank is the Biophysical Chemistry group, including members past and present: Dale Ang, Glen Dorrington, Claire Dow, Shirin Jamshidi, Joe Jones, Maria Lizio, Daniella Lobo, Kat Lloyd, Alix Martin, Caroline Montgomery, Steve Norton, Ben Pages, Marco Pinto Corujo, Kasra Razmkhah, Meropi Sklepari, Tamara Smidlehner, James Teahan and Alan Weymss. Throughout my project, each one has contributed ideas and suggestions to my work at some point and I would not have been able to get to this stage without them.

Finally, I am extremely grateful to the academic staff who have guided my project. Dr. Vasily Kantsler for his contributions and suggestions to the microfluidics side of my project; the late Prof. Mark Rodger, who offered assistance with my Fortran woes; Prof. Alison Rodger for her gentle nagging and pastoral support as well as her invaluable knowledge of biophysical chemistry; and last but not least, Dr. Nikola Chmel, for his guidance, support and patience (and he really did need lots of patience!) in helping me with this project.

Declaration

To the best of my knowledge, the material contained in this thesis is my own work except where otherwise indicated, cited, or commonly known. The Fortran code detailed in the appendices is based on code written and provided by Prof. Ravi Jagadeeshan, and is based on algorithm published by Prabhakar and Prakash [112]. Modifications made to the original code are detailed in Chapter 3 and the Appendices. The material in this thesis is submitted to the University of Warwick in partial fulfilment for the degree of Doctor of Philosophy in Mathematical Biology and Biophysical Chemistry, as described in the MOAC course regulations. It has not been submitted to any other university or for any other degree.

Abstract

Linear dichroism (LD) spectroscopy is a technique which uses polarised light to make inferences on the relative orientations of chemical groups within a molecule. A requirement for experiments using this technique is the alignment of molecules in the analyte. This thesis considers the use of fluid flows to align molecules.

The use of bead-spring and bead-rod models to simulate the behaviour of polymers and particles in fluid flows has been extensively discussed in the literature. In this work, the FENE-Fraenkel spring force law is implemented in bead-spring chain simulations, to assess the alignment behaviour of molecules in flow. Analytical properties of FENE-Fraenkel springs are derived and used to inform the coding of the simulation programme and to assess its outputs. The results of these simulations were comparable to both analytical results in zero-shear conditions and to data previously published by Hsieh et al. [48] on bead-spring chains in extensional flow, albeit with a reduced signal-to-noise ratio. Simulations of the alignment of FENE-Fraenkel spring dumbbells in shear flows show increased alignment with high shear rates, similar to modelling data published by McLachlan et al. [89].

This work also considers the use of microfluidic devices manufactured from polydimethylsiloxane to perform linear dichroism studies on DNA and vesicles in solution. A wide channel and cross-slot device are developed to provide conditions for pressure-driven and extensional flow to align molecules. Both devices are shown to align DNA molecules to measure an LD signal, with the cross-slot performing better than the wide channel device. With solutions of vesicles containing 1,6-Diphenylhexa-1,3,5-triene, a membrane probe, the wide channel device is shown to provide weak vesicle alignment to measure an LD signal. However, both microfluidic devices are unable to align molecules as well as Couette flow, the method normally used to align molecules for LD experiments.

Chapter 1

Introduction

Spectroscopic analysis is an invaluable tool in studies of the structure and functions of biomolecules. Visible and ultra-violet (UV) spectroscopy is a commonly used technique and, in the last century, specialised methods have been particularly useful in analysing properties such as the structural composition of proteins and the binding of molecules in a system. Linear dichroism (LD) is a particular variant of UV spectroscopy which involves the use of linearly polarised light.

LD experiments have most notably been used for the study of deoxyribonucleic acid (DNA) molecules, and DNA binding agents to determine the method of binding. Experiments have also been carried out to study soluble and membrane bound peptides, informing us of the relative orientation of elements of secondary and tertiary structure within them. A key part of performing an LD experiment is a requirement to align the molecules in a sample; the most common method of aligning molecules in a solution is to apply shear flow.

Shear flow LD experiments have been performed on solutions of polymers and long, rod-like molecules, including DNA, bacterial divisome FtsZ peptide filaments and bacteriophage. The behaviour of such molecules in fluid flows has been studied previously, using simulations with so-called bead-rod and bead-spring chains. More recently, work has been done to try to develop bead-spring chain simulations to more accurately mimic the behaviour of polymers in real life. In this thesis, we consider one such development; a rigid spring, known as FENE-Fraenkel, which combines a finite extensibility and com-

pressibility (spring-like qualities) with a non-zero natural length (rod-like qualities).

LD experiments have also been performed with membrane-bound peptide domains. Their structures are notoriously difficult to study outside the lipid-membrane environment and the use of solutions containing lipid vesicles with bound peptides makes LD a versatile technique for peptide structural studies. It is demonstrated in the literature that the vesicle elongation that provides for alignment in LD experiments when shear flow is used, can also be obtained by using extensional flow. This thesis also aims to assess if such a flow setup is viable for LD experiments with membrane-bound peptide domains using microfluidic devices.

1.1 Absorbance and linear dichroism

Spectroscopy can be defined as the interaction of electromagnetic radiation with matter [8]. This work is concerned with absorbance (or absorption) spectroscopy, where the focus is on the wavelengths and amounts of radiation that are absorbed by molecules.

A molecule can attain different energy states and levels in different forms; these include rotational, vibrational and electronic. When a molecule absorbs electromagnetic radiation, the energy from that radiation raises the molecule to a higher energy state. The amount of energy absorbed corresponds to the frequency of the radiation absorbed. This relationship is given by the *Planck-Einstein relation*:

$$\Delta E = h\nu = \frac{hc}{\lambda} \tag{1.1.1}$$

where ΔE is the energy absorbed, ν is the frequency of the radiation, c is the speed of light, λ is the wavelength of radiation and h is *Planck's constant* [8]. The energy states that a molecule can occupy are discretised; specific amounts of energy are required for a molecule to transition between two different energy states. So specific energy transitions are made depending on the particular wavelength of radiation being absorbed.

An example of absorbance in action takes place in photosynthethic reactions in plant leaves. Chlorophyll molecules absorb light waves with wavelengths in the ranges of 430 nm – 480 nm and 650 nm – 680 nm [9]. This radition excites the electrons in the

molecule to a higher state and allows them to be passed on electron acceptor molecules. These events form part of the electron transport chain in the light-dependent reactions of photosynthesis. Another example of absorbance is the medical use of X-ray radiation (wavelength range: 10 nm – 100 pm); the aborbance of electromagnetic radiation is strongly dependent on the atomic number of atoms present, and the high density of calcium in bones means that X-rays are absorbed more strongly by bone than by flesh.

In this thesis, the focus is on the ultraviolet and visible light regions of the electromagnetic spectrum (wavelength range: 100 nm – 1 μm). The energy transitions that correspond to absorbance of radiation in these frequency ranges are usually electronic; the electrons in the bonds of a molecule change energy states [8].

Absorbance is calculated by taking a logarithm of the ratio of the light intensities before and after it has passed through a sample. This value is also directly proportional to the concentration of molecules in the sample and the distance through which the light travels through the sample [97]. It is known as the *Beer-Lambert Law*:

$$A_\lambda = \log_{10} \frac{I_0}{I} = \varepsilon_\lambda C l \tag{1.1.2}$$

where:

- A_λ is the absorbance of the sample for light of wavelength λ.
- I_0 is the initial intensity of light before it enters the sample.
- I is the light intensity after it has gone through the sample.
- ε_λ is the *extinction coefficient*. This is specific to the molecule in the sample, the environment that they are in (*e.g.* solution they are in) and the wavelength λ.
- C is the concentration of the light-absorbing molecules in the sample.
- l is the *path length* — the distance that the light passes through the sample.

Electromagnetic radiation is composed of oscillations in an electric field and a magnetic field. These oscillations are perpendicular to each other and to direction of propagation. They are usually viewed as transverse waves travelling through space, and these waves occupy a particular plane or *polarisation*. For simplicity, only the electric component of an electromagnetic wave is considered here. In most circumstances, the light from

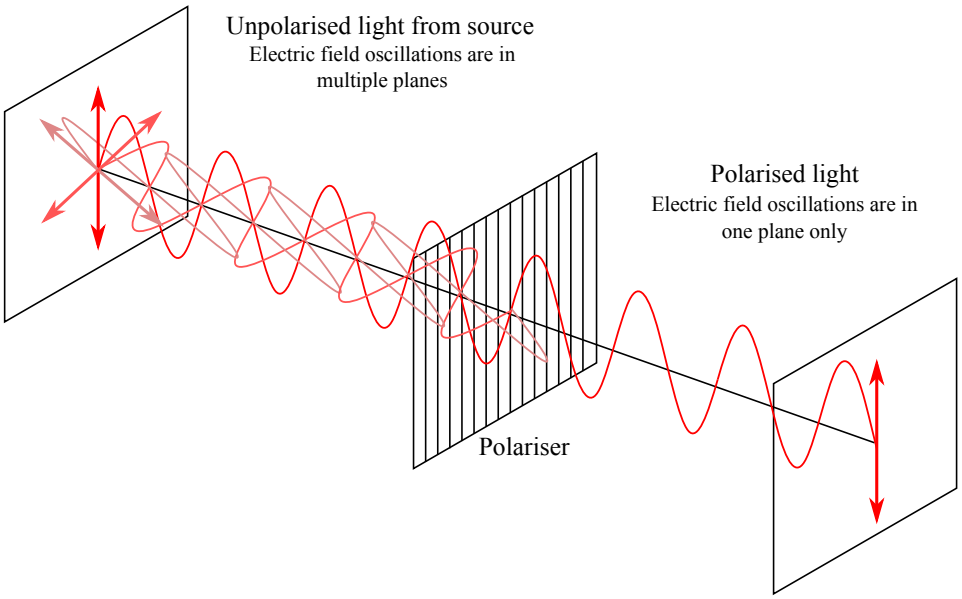


Figure 1.1: An illustration of unpolarised and polarised light. A polariser is used to allow light of a particular polarisation to pass through (in this case, vertically polarised light); all other polarisations are blocked.

a light source is unpolarised; the waves emanating from the source are not travelling in the same plane. The electronic transitions in a molecule that give rise to molecular absorbance phenomena are not isotropic; they are vectors with a specific direction as well as a specific energy. In order for electrons in a particular transition to be excited by electromagnetic radiation, the photon must not only have the correct energy, but also a polarisation component corresponding to the transition vector direction.

Linear dichroism is a specialised form of absorbance spectroscopy which makes use of this feature of electronic transitions. In absorbance spectroscopy, light with multiple polarisations shines on a sample and any electronic transitions, irrespective of their direction. In linear dichroism, the light is linearly polarised; that is, the light waves travel in one plane rather than in multiple different planes. So, provided that the molecules in a sample are aligned in some form, electronic transitions in the polarisation direction will absorb energy preferentially.

A linear dichroism spectrum is obtained by aligning the molecules in a sample, taking an absorbance spectrum using light polarised parallel to the alignment direction, taking another spectrum with light polarised perpendicular to the alignment direction, and finally

subtracting the perpendicularly-polarised light spectrum away from the parallel-polarised light spectrum. This is summarised by Equation (1.1.3).

$$LD = A_{\parallel} - A_{\perp} \tag{1.1.3}$$

An electronic transition parallel to the direction of molecular alignment will give a positive LD signal; similarly, if the transition is perpendicular, a negative LD signal will be observed.

Electronic transitions are seldom perfectly perpendicular or parallel to an alignment direction; but this does not affect the LD measurement. It can be shown that the strength of the LD signal is dependent on the angle the electronic transition makes with the alignment axis [97, 98]. Furthermore, the LD signal also depends on the proportion of molecules that are aligned in the desired way. The LD signal is numerically characterised by Equation (1.1.4) [97, §6.2].

$$LD = \frac{3}{2} S A_{\text{iso}} \left(3\cos^2\alpha - 1\right) \tag{1.1.4}$$

where:

- S is the *orientation parameter*, a number between 0 and 1 indicating the proportion of molecules oriented in the desired alignment direction.
- A_{iso} is the *isotropic absorbance* — the absorbance measured when the molecules in the sample are not aligned.
- α is the angle of the electronic transition with respect to the molecular alignment direction.

Thus, whereas traditional absorbance spectroscopy can be used to tentatively identify structures through the presence of characteristic electronic transitions, linear dichroism experiments can also identify the *relative orientation* of these transitions and the structures to which they belong. This orientational data can be isolated by calculating the *reduced LD*, by dividing the LD signal value by the isotropic absorbance [97].

$$LD^r = \frac{LD}{A_{\text{iso}}} = \frac{3}{2} S \left(3\cos^2\alpha - 1\right) \tag{1.1.5}$$

Note that this removes any dependence on conentration or path length contained in the absorbance (by the Beer-Lambert law).

However, to obtain distinctive peaks in an LD (and in a reduced LD) spectrum, the orientation of the molecules in a sample is crucial. If the proportion of chromophores (the parts of a molecule with electronic transitions that absorb light) aligned in the same direction is too low, then the signal will not be distinguishable from noise. The developement of molecular alignment methods is therefore an important part in the development of linear dichroism spectroscopy as an analytical technique.

A number of different methods have been developed to orient molecules in a variety of situations [15], including gel orientation [155], electric and magnetic fields [97] and drying samples on stretched films [86, 105]. This thesis focuses on aligning molecules in solution using shear flow.

For rigid or semi-flexible polymers, such as DNA, in solution, it is thought that flowing such a solution past a stationary surface at $0.1\,\mathrm{m\,s^{-1}}$ to $3\,\mathrm{m\,s^{-1}}$ results in a net orientation of the long axis of the molecules in the flow direction. This is due to the shear forces of the fluid flow acting on the molecules [97]. If light is passed through the sample perpendicular to the flow direction, then an LD signal from the fluid sample can be measured.

A design of a closed system to flow fluid samples between the walls of two concentric cylinders was developed by Wada [157, 158] based on the Couette shear flow (also known as Taylor-Couette flow).[1] Our lab has developed this initial device into a setup, summarised in Figures 1.2 and 1.3. A small volume, roughly $70\,\mu\mathrm{l}$, of fluid sample is placed in a rotating quartz cuvette, with a stationary quartz rod placed in the centre [82, 83]. The temporally polarised light beam passes through a focussing lens, through the cylindrical cuvette setup and then into a photomultiplier tube for detection.

This cell aligns molecules in a sample through the use of shear flow. As an example, polymer molecules in shear flow align with the long axis parallel to the shear axis, and therefore perpendicular to the radius of the cuvette.

Typically, the cuvette is rotated at $\omega = 3000\,\mathrm{rpm} = 100\pi\,\mathrm{rad\,s^{-1}}$. This gives rise to a shear rate of $\dot{\gamma} = 600\pi\,\mathrm{s^{-1}} \approx 1900\,\mathrm{s^{-1}}$. Furthermore, if the cylinders can be assumed to be of

[1] Couette flow is named after Maurice Frédéric Alfred Couette, but independent investigations into this type of flow were also carried out by Henry Reginald Arnulph Mallock, Sir George Gabriel Stokes and Sir Geoffrey Ingram Taylor [23, 80, 81, 122, 141, 149]

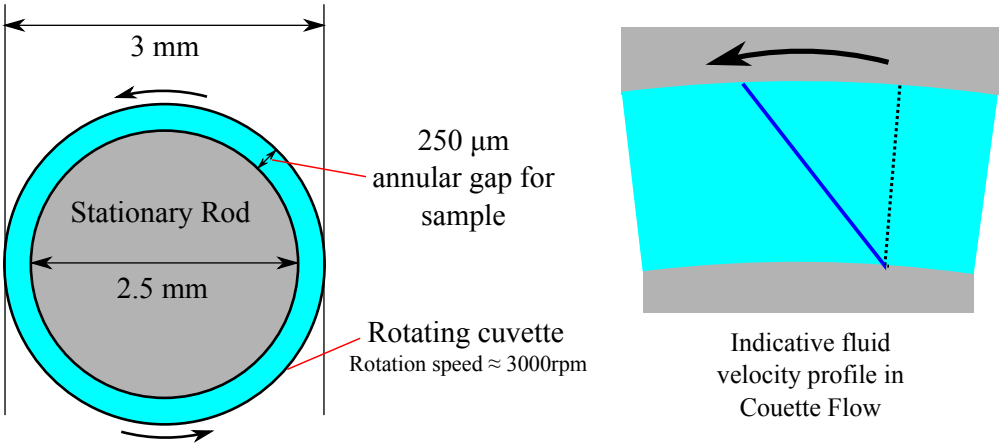


Figure 1.2: Dimensions of the Couette flow cell used for LD experiments.

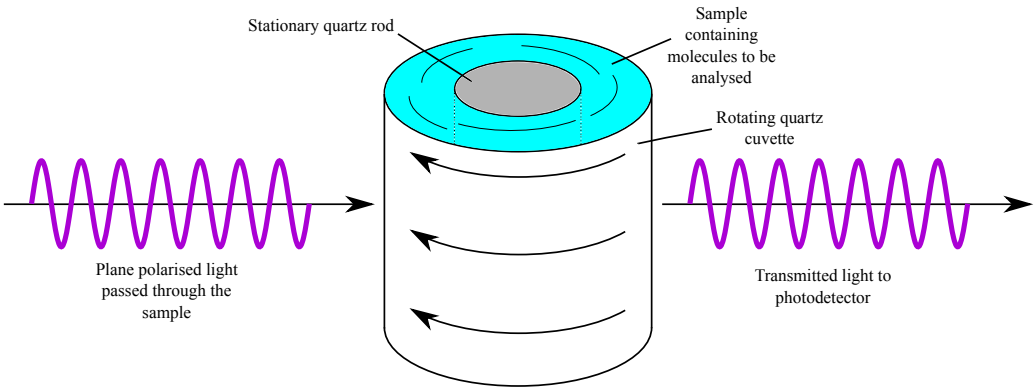


Figure 1.3: Schematic of a Couette flow cell for LD experiments.

infinite length, the velocity of the fluid at a distance r from the central axis is [149]:

$$v_\theta(r)/\mathrm{m\,s^{-1}} = \frac{3600\pi}{11}\left(r - \frac{1.5625\times10^{-6}}{r}\right)$$

This gives an almost linear shear profile between the stationary inner rod and the rotating outer cuvette, with speeds varying between zero at surface of the stationary rod, to $0.47\,\mathrm{m\,s^{-1}}$.

It is worth noting that this solution is not correct when the cylinders are taken to have a finite length; the velocity profile would also depend on the z-position (i.e. position along

the cylinder length). A general solution in such a situation can be derived using an integral transform method and involves Bessel functions of the first and second kind [162]. Such a solution is cumbersome. However, in situations where the aspect ratio (the ratio between the cylinder length and the annular gap size) is high and the radius ratio (the ratio between the annular gap size and the inner cylinder radius) is low, it can be shown that the flow profile in the case of infinite cylinder lengths can be used to reasonably approximate the velocity profile in the finite cylinder length case [162].

The Reynolds number in this cell is:

$$Re = \frac{\rho \omega R d}{\mu} \approx 132$$

where:

- ρ is the fluid density. This is assumed to be the density of water, $1000\,\mathrm{kg\,m^{-3}}$.
- ω is the cuvette rotation speed, $100\pi\,\mathrm{s^{-1}}$.
- R is the (inner) radius of the cuvette, 1.5 mm.
- d is the size of the annular gap, 250 µm.
- μ is the dynamic viscosity of the fluid. This is assumed to be that of water at 25 °C, $8.9 \times 10^{-4}\,\mathrm{kg\,m^{-1}\,s^{-1}}$.

Schultz-Grunow [132] determined that, for a Taylor-Couette flow system with a rotating outer cylinder and stationary inner cylinder, the critical Reynolds number (above which turbulent flow with Taylor vortices can be observed [149]) is of the order of 10^3 [67]. This would indicate that the flow in the cell used for linear dichroism experiments is laminar, stable and axisymmetric. However, research by DiPrima et al. [28] indicates a critical Reynolds number of around 100 for the cell with a radius ratio of 0.83, where the inner cylinder is rotating and the outer cylinder is stationary. This would suggest that the flow in the cell used for linear dichroism experiments is turbulent. However, it is important to note that the values calculated by DiPrima et al. [28] are based on a rotating *inner* cylinder rather than an outer cylinder. Furthermore, for vesicle solutions, it can be assumed that the fluid viscosity is similar to that of water. However, with solutions of long chain molecules which interact with one another, the viscosity is higher; DNA solutions, for example, can

have dynamic viscosities of around $1.2 \times 10^{-3}\,\mathrm{kg\,m^{-1}\,s^{-1}}$ [40]. The increase in viscosity results in a lower Reynolds number, making a stable axisymmetric flow more likely.

1.2 LD experiments with polymers in solution

Linear dichroism experiments have been used extensively to study biopolymer molecules [122]. One of the earliest experiments was conducted by Cavalieri et al. [22], where the dichroism[2] was measured of DNA solutions in flow using a rectangular quartz cell. They demonstrated that pH and salt concentration affect both the viscosity of the solution and the dichroism recorded, indicating a link between the macrostructure of DNA solutions and the dichroism. These results were extended by Rizzo and Schellman [119], with investigations into the effect of salt concentrations on the flow dichroism of DNA from T7 bacteriophages, using a Couette flow cell instead.

The basis for the LD signal observed from solutions of aligned DNA molecules is in the electronic transitions present in the bases [24, 122, 157]. These transitions lie in the plane of the bases [85, 96]; therefore, when DNA molecules are aligned, light polarised in the direction of the molecule axis should not excite electrons in these dipole moments, whereas light polarised perpendicular to the aligned molecules will. The result is a strong LD signal correspoding to electronic transitions perperndicular to the molecule at around 260 nm.

The strength of this signal has been used to measure the interaction of compounds with DNA molecules. For example, Rodger et al. [120] used the reduction of LD signal strength from DNA to monitor the structural disruption caused by Cobalt (III) Tetra- and Pentamines. The majority of examples in the literature on the applications of LD to structural studies of biomacromolecules involve DNA and DNA-ligand systems [24]; particularly in the development of anti-cancer drugs, where compounds are developed to target DNA in tumour cells to prevent their replication. There are a number of examples of the use of LD to monitor the binding of compounds in the minor groove [94], in the major groove [73, 91] and between bases [105, 156] of DNA molecules.

[2] The *dichroism* is closely related to, but not the same as, linear dichroism.
It is defined by $D = (A_{\mathrm{iso}} - A_{\parallel})/A_{\mathrm{iso}}$ [22].

Fibrous proteins have also been studied with linear dichroism spectroscopy. Existing techniques such as X-ray crystallography and nuclear magnetic resonance spectroscopy (see below) can be used to provide structural information on the monomers that make up protein fibres, but are not able to provide as much information on the polymer fibres structures themselves. Linear dichroism spectroscopy provides a means of studying the orientation of monomers within protein fibres, and molecules which interact with the fibres [24]. An early example of this is work performed by Higashi et al. [45] and Miki and Mihashi [90] in the 1960s and 1970s on F-actin protein fibres (found in muscle tissue), and the binding of adenosine diphospahte to these fibres, using flow dichroism. In more recent work, Adachi et al. [3] used LD spectroscopy to study the formation of amyloid fibrils, which is known to be connected with conditions such as Alzheimer's disease, and Bulheller et al. [19] studied a range of proteins using LD, including FtsZ, a protein involved in bacterial cell division, and collagen, a protein found in connective tissues of organisms.

1.3 Simulating polymers in dilute solutions

A key part of peforming a linear dichroism experiment is finding a means to align molecules in the sample being studied. In a number of cases, including those mentioned above, when polymers are being analysed with LD, they are in an aqueous enviroment. So it is important to consider the behaviour of these polymers in fluid flows to understand how the alignment of these molecules can be achieved. Simulations of polymers in fluid flows have been used to study this behaviour. Rather than using a molecular dynamics simulation of a polymer constructed with specific atoms and bonds, simpler means of studying their behaviour have been developed. One of the simplest means of modelling a polymer is to use a chain of beads connected together; this forms the basis of all the models discussed below.

One of the earliest models of polymer behaviour in fluids was proposed by Rouse [124], where molecules were modelled using chains of beads and springs. The system was based upon Brownian dynamics, where the only forces acting upon the beads are random interactions between fluid molecules and the beads (simulated by a stochastic process),

and friction from the movement of the bead through the fluid [14, 154]. In addition, beads are also subject to tension forces from springs attached to them. In the Rouse model, these forces are proportional to the length of the spring; that is, the magnitude of the spring tension force, F, is

$$F = HQ, \tag{1.3.1}$$

where Q is the spring length and $H > 0$ is the spring constant. Springs with this property are said to be *Hookean*[3] [14]. Furthermore, in this model, the springs were allowed to freely rotate around the bead centres. This freely-jointed bead-spring chain model has

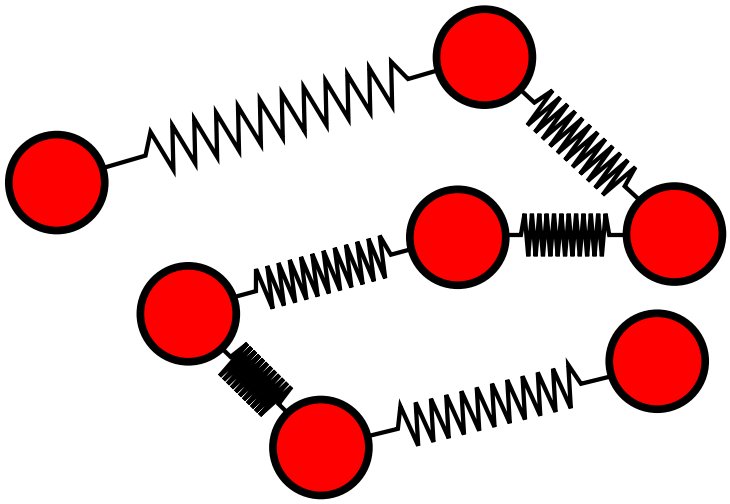


Figure 1.4: Example of a bead-spring chain with seven beads. In the Rouse model, these springs are Hookean (the tension force is proportional to their length) and freely-jointed to the beads (meaning that there is no restriction on the angle between two springs attached to the same bead).

formed the basis of models used in polymer physics simulations [14] and could begin to account for the elasticity seen in real life polymers that could not be modelled with chains of beads connected by rigid rods. The properties of simple bead-rod chain models were investigated by Kramers [68] in the 1940s, but these models have been used to study the behaviour of molecules in fluid flows. An early example is the study of bead-rod dumbbells (chains made up of 2 beads) in shear and extensional flows [13]. Bead-rod chain models have also been implemented in more recent work, such as that by Petera and Muthukumar [107] and Liu et al. [78], where external fluid flows (shear and elongational) are integrated into the model.

[3] named after English natural philosopher, Robert Hooke

Rouse's model was enhanced by the inclusion of *hydrodynamic interactions* (HI) by Zimm [169]. These are the indirect interactions between different parts of a molecule as a result of the behaviour of molecules within the surrounding solvent [66]. A number of different ways of implementing these interactions in a model have been proposed in the literature since, including by Öttinger [102], Rotne and Prager [123] and Yamakawa [165]. Furthermore, more recent work in the literature use models including *excluded volume* (EV) *interactions*, where a potential between any two beads provides a repulsive force between them. This concept, developed by Flory [36], is based on the principle that no two beads should ever occupy the same point in space.

The inclusion of these types of interaction have been implemented in models with bead-spring chains using Hookean springs [111]. However, the use of other spring force laws in bead-spring chain simulations has also been considered [154]. One force law commonly used in the polymer physics literature is the finitely-extensible non-linear elastic (FENE) spring. This force law was originally developed by Warner [161] and is based on the notion of there being a finite limit to the length of a spring, with the spring being stiffer as this length is reached. Expressing this mathematically, the magnitude of the spring tension force exerted by a FENE spring is

$$F = \frac{HQ}{1 - \left(\frac{Q}{Q_{\max}}\right)^2} \tag{1.3.2}$$

where Q is the length of the spring, H is the spring constant and $Q_{\max} > 0$ is the maximum length of the spring.

This behaviour is more closely related to that of real-life polymers than the Hookean spring force law, where there is no limit to the spring length. Indeed, studies that have used the FENE force law in simulations have reported closer correlation with experimental studies than those where Hookean springs were used [14, 35, 117, 154].

The FENE and Hookean spring models exert a non-zero tension force whenever they are of non-zero length. They are therefore quite different to bead-rod models, not just in the flexibility in the chain length, but also in there not being a natural, non-zero length between beads when there are no forces being exerted on them by connected springs or rods. An alternative spring force law was proposed by Fraenkel [37] to create a spring

with a non-zero natural length and where the tension or compression force is proportional to the deviation of the spring length from this natural length (the Hookean force law is similar but with an effective natural length of zero). The magnitude of the tension force, F, exerted by a Fraenkel spring is given by

$$F = H(Q - Q_0) \tag{1.3.3}$$

where Q is the spring length, H is the spring constant and $Q_0 \in [0, \infty)$ is the natural length of the spring.

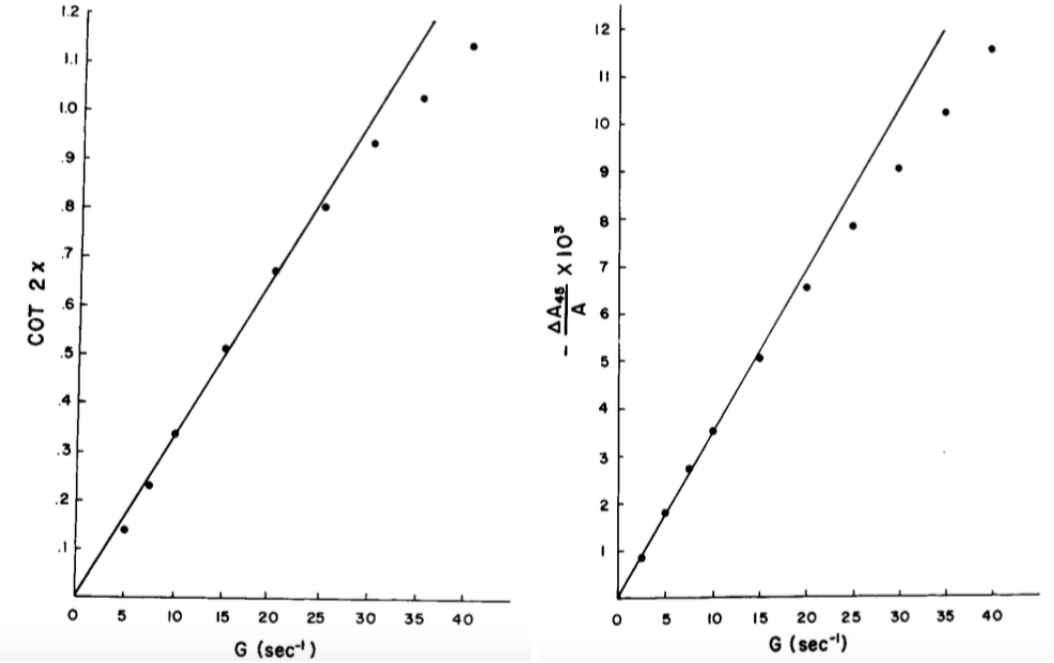
This force law allows for the spring to compress as well as stretch, and is also more similar to a rigid rod due to the stiffness provided by the non-zero natural length [14, 48, 163]. However, the use of the Fraenkel spring in computer simulations has been shown to require smaller time-step sizes than with bead-rod models, which means that simulations take longer to run and are computationally expensive [48].

To counteract this computational issue, and to try to incorporate the finite extensibility of the FENE force law that is missing in the Fraenkel law, Hsieh et al. [48] proposed a hybrid of the FENE and Fraenkel spring force laws, known as the FENE-Fraenkel force law. The proportionality between the tension/compression force and the deviation from the natural length is replaced by a non-linear relationship between the two where the spring is infinitely stiff when its length reaches a certain minimum or maximum. Computationally, the use of this spring force law allows for much larger time-steps than would be feasible with Fraenkel springs, but also allows for the inclusion of HI and EV interactions. These interactions are difficult to implement in bead-rod model simulations [48]. Mathematical details of the spring force laws mentioned here are provided in Chapter 2.

Alternative methods of bridging the gap between the computational performance of bead-rod and the versatility of bead-spring models have been considered by others [139]. One example is the use of succesive fine-graining, where simulations are run using a bead-spring chain with a fixed number of *Kuhn steps* (see below) and the model is progressively refined by adding more beads. This has been shown to provide excellent agreement with results for bead-rod and bead-spring models using the FENE-Fraenkel spring force law [108, 113, 143]. Further analysis of this method requires the discussion of other parameters in bead-rod/spring models, including the *persistence length*, the length of the chain

which is rigid enough to be treated as a rod, and the number of *Kuhn segments/steps*, parts of a chain which are rigid enough to be considered as rods. These concepts and their context in the study of DNA molecules and linear dichroism are covered in the literature, such as in work by Rittman et al. [118].

Comparisons between the alignment of DNA molecules in shear flow, experimentally derived using linear dichroism spectroscopy, and bead-spring/rod models have also been explored. The original flow dichroism study of DNA made reference to Rouse's bead spring model to develop predictions for the amount of dichroism to be observed [157]. Later research by Wilson and Schellman [164] concluded that their experimental results were comparable (but not in exact agreement) with predictions based on the Rouse-Zimm models, as shown in Figure 1.5.



(a) $\cot 2\chi$ vs G **(b)** $-\Delta A_{45}/A$ vs G

Figure 1.5: Graphs of $\cot 2\chi$ and $-\Delta A_{45}/A$ against G, where χ is the angle between the electronic transition of interest and the flow direction (corresponding to α in Equation (1.1.4)), A is the isotropic absorbance, ΔA_{45} is the change in absorbance observed between the flow direction and 45° to the flow direction and G is the shear rate. These experimental data were obtained by studying DNA in flow. The lines plotted are using linear least-squres regression. Theory from Rouse-Zimm models suggests a linear relationship between G and observable variables $\cot\chi$ and $-\Delta A_{45}/A$. This linear relationship is seen for low shear rates, but degenerates for values of G above $20\,\mathrm{s}^{-1}$. These figures are reproduced from Figures 1 and 2 of [164].

More recently, experimental work by Simonson and Kubista [138] looked at the alignment of DNA in Couette shear flow, with a focus on the effects of the length of DNA strands and the shear rate. This also concluded that, for low shear rates, predictions based on the

Rouse-Zimm models were borne out in experiments. Their analysis was based upon the use of the reduced LD (see Equation (1.1.5))[4] to isolate the orientation of DNA molecules. Furthermore, they derived the following expression for the orientation parameter S.

$$S = \left(\langle \cos^2\theta \rangle + \langle \cos^2\theta \cos^2\phi \rangle - \langle \cos^2\phi \rangle\right) \tag{1.3.4}$$

where

- θ is the angle between the x-axis (see Figure 1.6) and the local DNA helix axis.
- ϕ is the angle between the z-axis (see Figure 1.6) and the normal to the plane comprising the x-axis and the local DNA helix axis.
- $\langle \cdot \rangle$ mean that the bracketed value is averaged over all molecules.

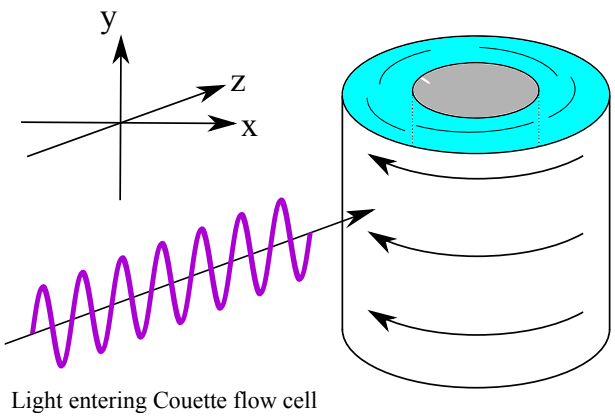


Figure 1.6: Diagram defining the axes referred to in the expression for the orientation parameter S (Equation (1.3.4)).

As a simplification, suppose that:

- The DNA molecules in the solution can be treated as dumbbells or rigid rods.
- Across all molecules, there is no dependence between θ and ϕ.
- All molecules are distributed in such a way that $\langle \cos^2\phi \rangle = 1/2$; that is, there is *uniaxial orientation*.

[4] While there are a number of electronic transitions in the plane of bases in DNA that give rise to absorbance signals around 260 nm, the assumption is of an effective angle, α_{eff} between a single representative electronic transition and the DNA molecule axis of roughly 86°, as established by Norden et al. [96].

Then the orientation parameter can be simplified to the following form [89, 97]:

$$S = \frac{1}{2}\left(3\langle\cos^2\theta\rangle - 1\right) \tag{1.3.5}$$

This simplification was used by McLachlan et al. [89] to consider the orientation of rigid rods of various lengths in shear flow to simulate DNA.

In this thesis, the aim is to make use of bead-spring models with the FENE-Fraenkel force law to obtain orientational information about polymers, such as DNA, in shear flow. This is done by modifying and applying a code used by Prabhakar and Prakash [111] to run bead-sping simulations, substituting FENE springs with FENE-Fraenkel springs, and calculating orientation parameter values for the molecules simulated. It is hoped that by obtaining data similar to that published by McLachlan et al. [89], that the program can be used to study chains of FENE-Fraenkel springs to study DNA alignment more accurately than before.

1.4 Studying the structure of membrane proteins

As well as polymers such as DNA and fibrous proteins, membrane proteins have also been studied with LD spectroscopy. Membrane proteins are biomolecules contained within the lipid bilayer membranes of cells. They are of significant interest because of the roles they play in a variety of processes in living things. These include cell signalling as apart of immune responses, control of substances entering and exiting a cell and monitoring and manipulating the shape of a cell. It is estimated that between 20% and 30% of the proteins in most organisms are membrane proteins. Furthermore, roughly 40% of drug targets are thought to be membrane proteins [9, 21, 47, 75, 103]. However, despite their importance, membrane proteins are difficult to study.

Cell membranes are typically made up of a bilayer of lipids molecules. Each of these molecules are made up of a hydrophilic head group and hydrophobic tail groups [16]. The natural grouping of the tail groups of multiple lipid molecules results in the formation of lipid bilayers (see Figure 1.7). The surface of most membrane proteins is relatively hydrophobic, due to the fact that most of their structure resides in the hydrophobic region

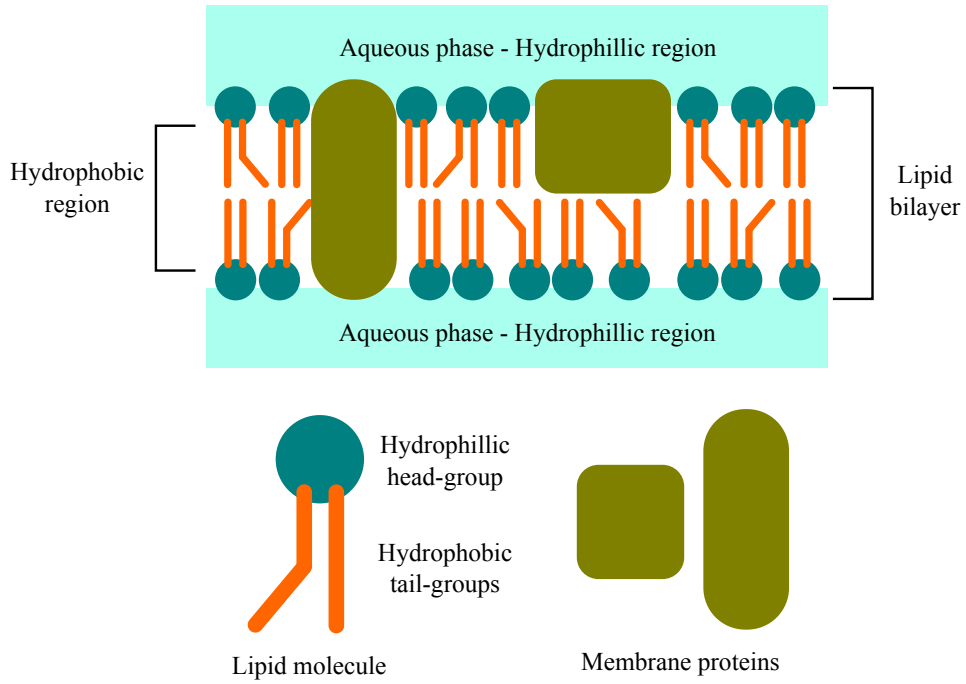


Figure 1.7: Illustration of the basic components of phospholipids and a schematic cross-section of a phospholipid bilayer with membrane proteins.

of the lipid bilayer. They therefore cannot be treated in the same way that other proteins, such as F-actin, are by studying them in aqueous solutions. Membrane proteins are also very sensitive to other features of their environment, such as pH, and changes in these factors will lead to structural changes in the protein. This means that the protein structure observed after extraction and purification may not reflect its native structure in the cell membrane. Proteins can be extracted from membranes with the use of detergents and studies suggest that they are likely to retain structural features in the process. However, protocols for the extraction and purification of these proteins from the detergent are non-trivial [20, 21, 126].

One starting point for establishing the structural content of a membrane protein (indeed, any protein) is the technique of circular dichroism (CD) spectroscopy. This uses circularly polarised light (as opposed to the linearly polarised light used in LD experiments) to identify structures based on their chirality (a form of molecular asymmetry) [97]. CD spectroscopy is concerned with the identification of protein secondary structure (see Figure 1.8). The chirality of α-helices and β-sheets can be detected with circularly polarised light, and thus the proportion of protein that is comprised of each of these elements of

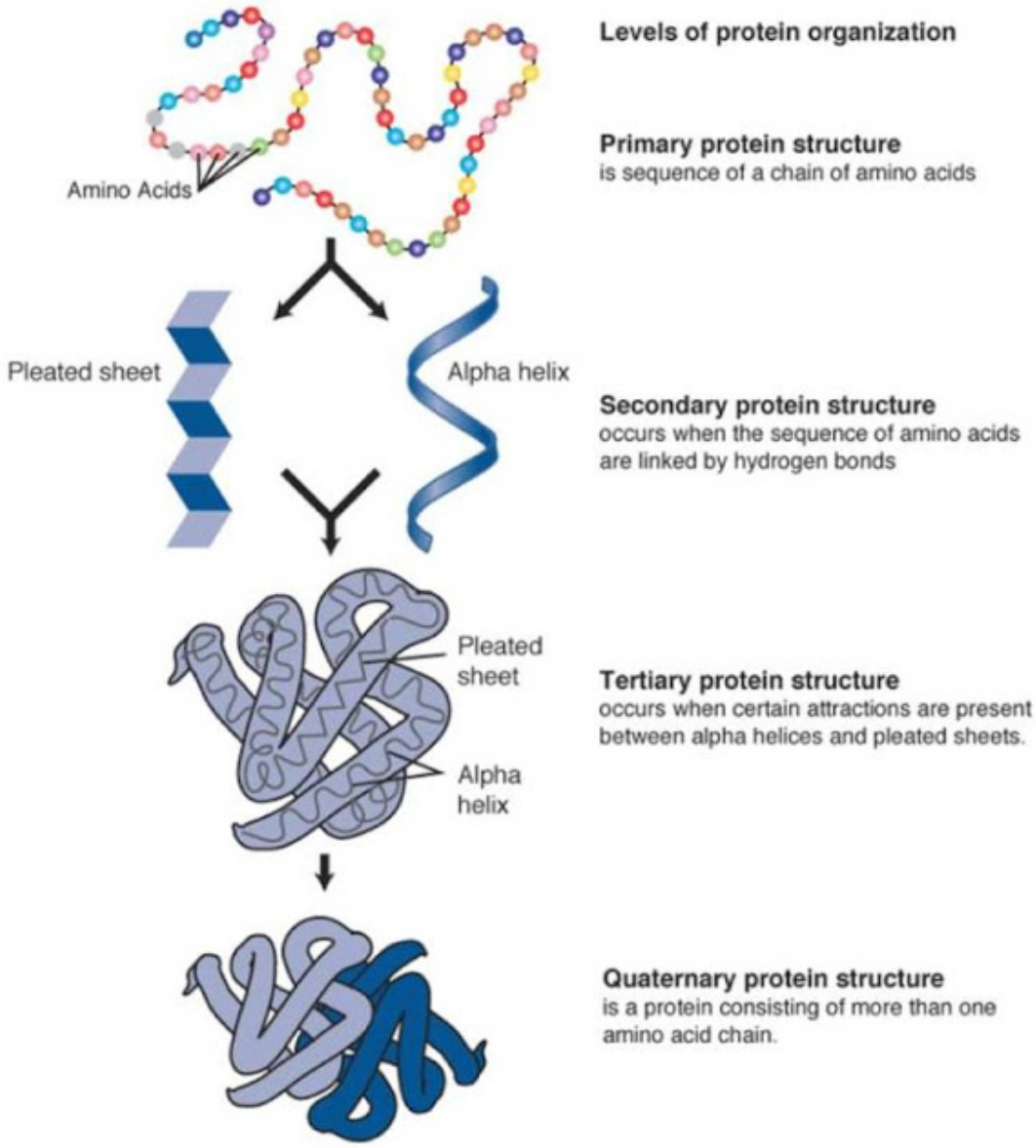


Figure 1.8: Illustration of the hierarchy of protein structure. All proteins are comprised of at least one chain of amino acid units (a *polypeptide chain*); most proteins are made up of multiple chains. The sequence of these amino acids in a chain is known as the *primary structure*. A polypeptide chain is usually folded up into a three-dimensional form, containing sub-structures known as α*-helices* and β*-sheets* held together by hydrogen-bonding; these make up the protein's *secondary structure*. Further folding and arrangements of the secondary structure gives rise to the *tertiary structure*. The collection of tertiary structures that make up a protein are known as the *quartenary structure*. A more detailed treatment of protein structure can be found in [75].

secondary structure can be determined. However, it does not provide information about the location of these sub-structures within the protein [97].

A technique that can resolve the molecular detail of proteins more thoroughly is X-ray crystallography. This involves the formation of crystals made up of the protein. X-rays are then passed through the crystals to give a diffraction pattern. This pattern provides information about the relative location of atoms within a protein molecule [30]. While this technique has proven to be effective at elucidating the molecular structure of some proteins, the major bottleneck is the formation of crystals, and membrane proteins are notoriously difficult to crystallise [20]. Nuclear magnetic resonance (NMR) spectroscopy is an alternative technique that uses the magnetic properties of the nuclei of specific elements to generate structural information about a molecule. This technique has been used to derive the structure of roughly 80 membrane proteins, and has also been used to obtain information about the motions of and interactions between substructures within a protein. This technique is limited by the large size of membrane proteins, though recent work has shown that structural studies of membrane proteins as large as 30–50 kDa are possible with NMR [62]. Furthermore, the sensitivity of membrane proteins to their environment means that the preparation of samples for NMR spectroscopy is a major factor in performing such a structural study.

With these concerns, there is a preference to study membrane proteins in their native environment to avoid changes in structure. However, there is a difficulty in studying the membranes of active cells, as there are other chemical species present that would interfere. Instead, proteins can be embedded in the membranes of *vesicles*, simple fluid pockets bound by a lipid bilayer, where the membrane is similiar in lipid composition to that in which the protein is usually found. Furthermore, given the sensitivity of membrane proteins to their environment, indirect methods to study their behaviour have also been considered. One example in the literature of this is the GALLEX assay devised by Schneider and Engelman [130]. This was used to study the interactions between α-helices in the membranes of *E. coli* cells. Modified forms of the transmembrane domain peptides were constructed, so that the helices of concern were unnaffected but had an extra sequence of amino acids attached. The interaction of the helices would result in the binding of two of the attached sections, which would result in a chain of biochemical processes affecting the production of a chemical. The interactions between the transmembrane domains could

therefore be monitored by measuring the concentration of this chemical [130]. While this approach was shown to work in this instance, such an approach may not be suitable for other membrane proteins where it is not possible to add extra amino acids to construct this indirect monitoring mechanism.

Linear dichroism spectroscopy is a technique that could be used to study membrane proteins in their native environment [98] and potentially allow for studies similar to the GALLEX assay, but without the complication of modifiying proteins. For example, if an interaction between two transmembrane domains causes a change in their orientation in the membrane, then this interaction could be measured directly with LD spectroscopy, rather than indirectly as with the GALLEX assay. Furthermore, this has the advantage of providing a larger structure that can be controlled to give the alignment desired; put simply, if we wish to orient the protein, we need to orient the membrane containing it. This was proven with the use of aromatic molecules embedded in lipid membranes of vesicles by Ardhammar et al. [5]. Finally, the electronic transitions present in protein secondary structures are known and can be used to gain data of the relative orientation of these structures within a protein [5, 24, 97, 121]. A number of studies of membrane proteins with LD spectroscopy have been documented in the literature. For example, Schoepp et al. [131] used squeezed gels to align membranes with cytochrome b_6f to study the spatial organisation of iron-containing complexes within the protein, and Couette shear flow systems were used by Rodger et al. [121] to study gramicidin, cytochrome *c*.

1.5 Aligning vesicles in fluid flow

As previously mentioned, molecular alignment is crucial in LD spectroscopy. The examples mentioned above refer to two different alignment methods: using a gel squeeze and using shear flow to align vesicles with membranes containing the protein of interest. This thesis shall focus on vesicles in fluid flows.

Lipid vesicles can be categorised according to their size and lamellarity (if a vesicle is contained within a larger vesicle). The main categories, shown in Figure 1.9, are small unilamellar vesicles (SUVs), medium unilamellar vesicles (MUVs), large unilamellar vesicles (LUVs), giant unilamellar vesicles (GUVs) and multilamellar vesicles (MLVs) [55, 79].

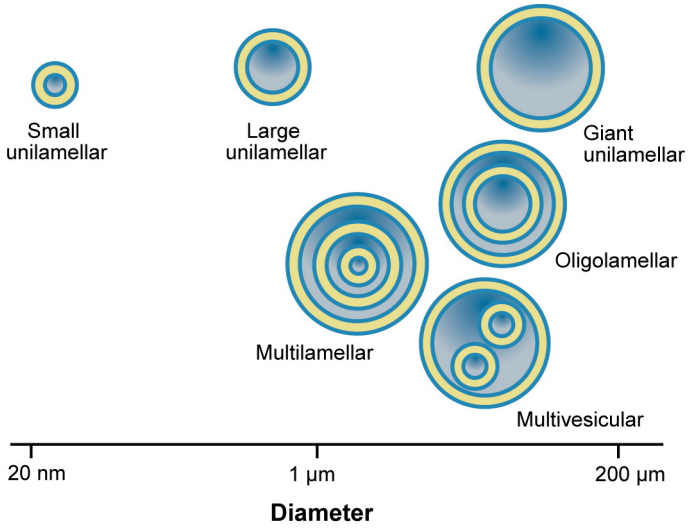


Figure 1.9: Vesicle types based on diameter and lamellarity. This diagram is reproduced with permission from the Figure 2 of [55].

When carrying out experiments with membrane proteins in vesicles, the strucutral integrity of the protein is usually verified using CD spectroscopy before use in further experiments. To avoid errors in the measurement due to light scattering by large particles, one of the requirements of this technique is that the vesicles sizes are as small as possible to avoid false measurments due to light scattering by large vesicles. Typically vesicle diameters should be no greater than 400 nm, implying that MUVs should be considered [160]. However, there is very little research in the literature on the dynamics of vesicles of that size; a large proportion of research into vesicle dynamics in fluids is focussed on

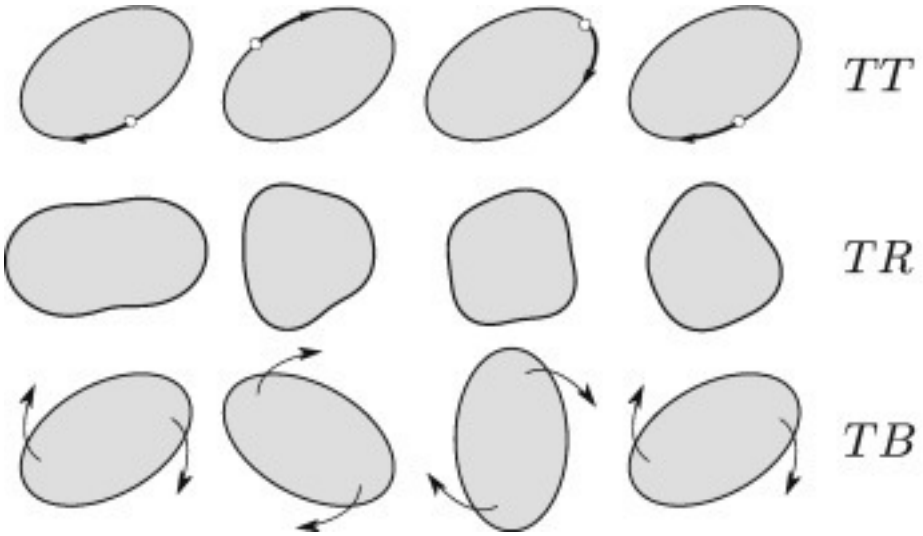


Figure 1.10: An illustration of the three types of behaviour that vesicles undergo in planar fluid-flow. The three lines corresponds with (from top to bottom) tank-treading (TT), trembling (TR) and tumbling (TB). These diagrams are reproduced with permission from the abstract figure of [2].

GUVs. Abreu et al. [2] published a review summarising this research, and the reader is asked to consult their work for full details. This section focuses on some of the major concepts discussed in the review paper.

In planar fluid flows, the three vesicle behaviours that are observed in fluid flows restricted to one plane are:

- *Tank-treading*, where the membrane rotates around the fluid bulk and the vesicle maintains the same angle with respect to the external environment.
- *Tumbling*, where the whole vesicle body rotates around its centre.
- *Trembling*, where the vesicle membrane fluctuates in an uncontrolled way.

These are illustrated in Figure 1.10. These three behaviours have been shown to take place in experimental conditions [2, 26, 58, 64, 76], but there have also been studies published to develop the theory behind what causes these three behaviours. The three main parameters that are thought to play a role in a vesicle's behaviour in planar flows are:

- The *excess area*, Δ, corresponding to the deformation and stretching of the vesicle membrane in flow. This is defined by:

$$\Delta = \frac{A}{R_0^2} - 4\pi \tag{1.5.1}$$

where A is the surface area of the vesicle and R_0 is the *effective radius* of the vesicle, given by

$$R_0 = \left(\frac{3V}{4\pi}\right)^{1/3}. \tag{1.5.2}$$

i.e. the vesicle's effective radius is the radius of a sphere with the volume of the vesicle.

- The *viscosity contrast*, λ, equal to the ratio of the fluid viscosities inside and outside the vesicle, denoted η_i and η_o respectively.

$$\lambda = \frac{\eta_i}{\eta_o} = \frac{\text{Fluid viscosity inside the vesicle}}{\text{Fluid viscosity outside the vesicle}}. \tag{1.5.3}$$

- In the case of linear shear flows, the *capillary number*, defined by

$$Ca = \frac{\dot{\gamma}\eta_o R_0^3}{\kappa}, \tag{1.5.4}$$

where $\dot{\gamma}$ is the shear rate, η_o is the viscosity of fluid outside the vesicle, R_0 is the effective radius as given by Equation (1.5.2) and κ is the *membrane bending energy*. This is a dimensionless number that characterises the ease with which the viscous forces of the surrounding fluid will overcome the rigidity of the vesicle membrane to alter the vesicle shape. As an example, let: $\dot{\gamma} = 1900\,\mathrm{s}^{-1}$, as in the case of the Couette flow cell described above; $\eta_o = 8.9 \times 10^{-4}\,\mathrm{kg\,m^{-1}\,s^{-1}}$, the dynamic viscosity of water at 25 °C;

- $R_0 = 50\,\mathrm{nm}$, the radius of a vesicle typically studied in CD and LD experiments; and

- $\kappa \approx 25 k_B T$, as suggested by Abreu et al. [2], with $k_B = 1.38 \times 10^{-23}\,\mathrm{J\,K^{-1}}$ as Boltzmann's constant and $T = 298\,\mathrm{K}$. In this instance, the capillary number is $Ca = 0.0021$.

For more general planar fluid flows which are not necessarily of the linear shear type, these can be broken down into extensional (or elongational) and rotational components, as illustrated in Figure 1.11.

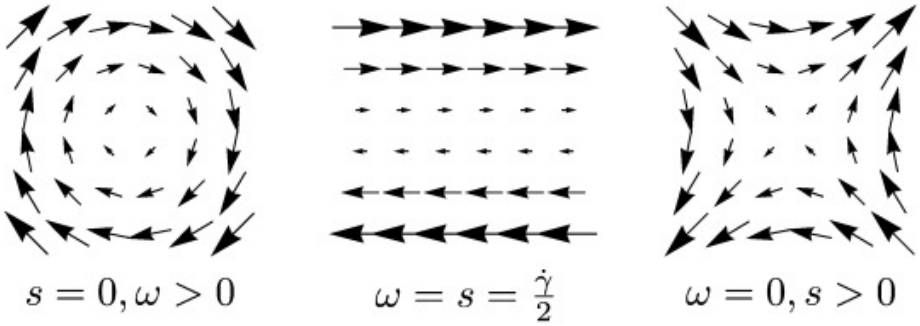


Figure 1.11: A diagram illustrating the rotational (left) and extensional/elongational (right) components of a linear shear flow (middle). Reproduced from Figure 2 of [2].

Mathematically, the relationship between the fluid velocity vector, $\mathbf{v}$, and the parameters ω and s is given by

$$\mathbf{v} = s(y\mathbf{e}_x - x\mathbf{e}_y) + \omega(y\mathbf{e}_x + x\mathbf{e}_y),$$

where $\mathbf{e}_x$ and $\mathbf{e}_y$ are the basis vectors for the x- and y-axes respectively. If these are equal, the result is a linear shear flow, and $\omega = s = \dot{\gamma}/2$. For more general planar fluid flows, ω and s, the parameters corresponding to the extensional and rotational components of the flow, are considered instead of $\dot{\gamma}$.

The exact details of the parameter values at which changes between the three different vesicle behaviours occur have been the subject of debate in the literature. A number of theoretical models suggest that three main parameters are required to plot the phase diagram between all the vesicle behaviours [11, 25, 34, 61, 166, 167]. However Lebedev et al. [71] postulated that the vesicle behaviour can be deduced by calculating the value of two parameters: S (not connected with the orientation parameter S used in the LD literature) and Λ defined below.

$$S = \frac{14\pi}{3\sqrt{3}} \frac{\eta_o R_0^3}{\kappa\Delta} s, \qquad \Lambda = \left(\frac{23\lambda + 32}{24}\right) \sqrt{\frac{3\Delta}{10\pi}} \frac{\omega}{s}, \tag{1.5.5}$$

where:

- R_0 is the effective radius (Equation (1.5.2)).
- Δ is the excess area (Equation (1.5.1)).
- κ is the membrane bending energy.

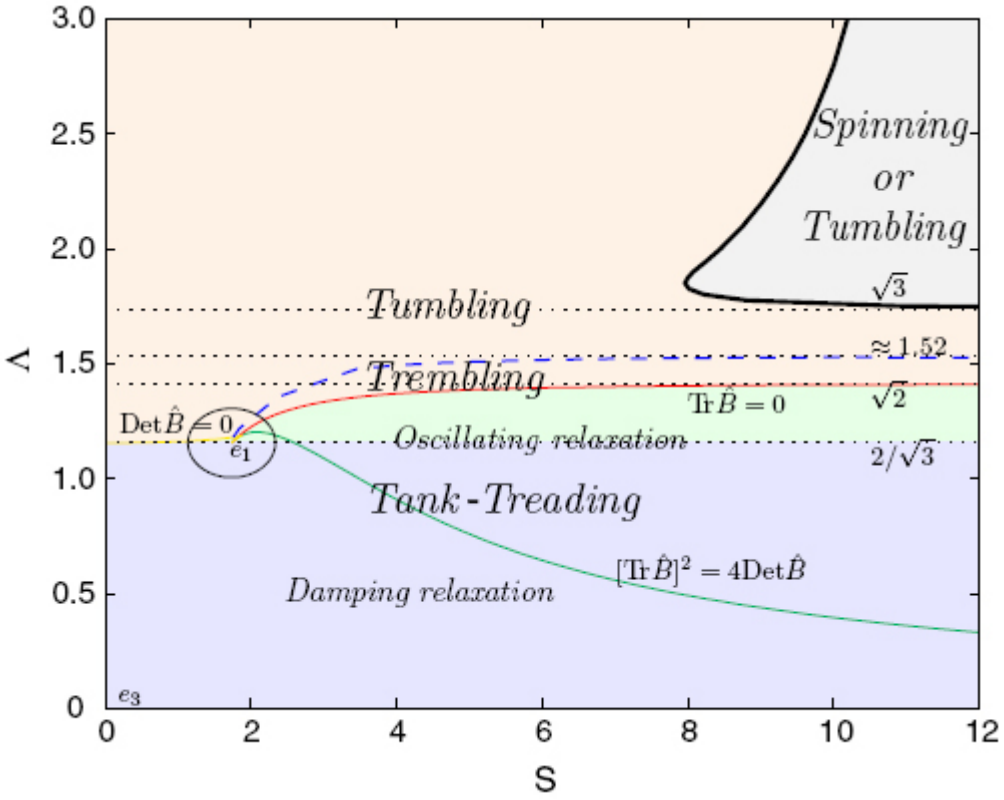


Figure 1.12: Phase diagram illustrating the regions of (S, Λ)-parameter space that correspond to the different vesicle behaviours. Tank-treading takes place in the blue and green regions, trembling in the area between solid red and broken blue lines and tumbling in the orange and grey areas. The diagram is reproduced from Figure 9 of [71].

- η_o is the viscosities of the fluid outside the vesicle.
- λ is the viscosity contrast (Equation (1.5.3)).
- ω and s correspond to the extensional and rotational components of velocity of the surrounding fluid.

The associated phase diagram for the vesicle behaviour with respect to these two parameters is shown in Figure 1.12. Experimental results suggest that this two-parameter setup is more closely followed by experimental results than the three-parameter setup proposed by other papers [2]. This setup is therefore considered in more detail below.

For linear dichroism experiments, it can be argued that tank-treading is the most preferred of these three behaviours. This is because neither tumbling nor trembling give rise to an environment where the vesicle's (and so the membrane's) alignment relative to the flow remains constant. Furthermore, since the membrane moves constantly around the vesicle in the tank-treading regime and provided that the vesicle is elongated in some way, the

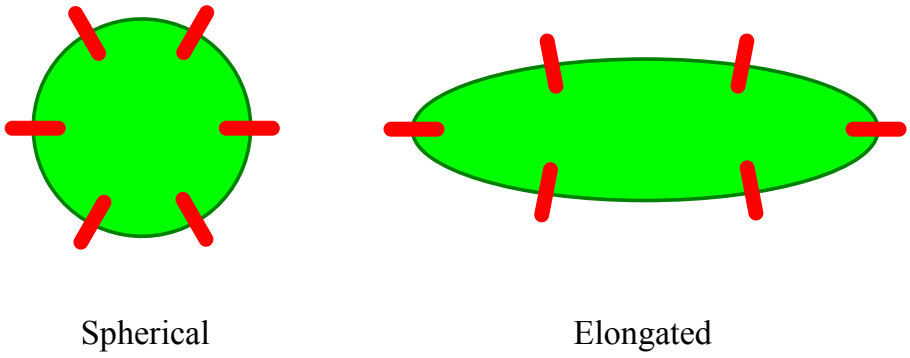


Figure 1.13: The elongation of a vesicle containing chromophores of interest in its membrane results in a net orientation of chromophores. In this example, the spherical vesicle has chromophores (in red) distributed evenly throughout its membrane, and the net orientation of the chromophores is zero. Elongation results in more of these chromophores occupying the parts of the membrane along the long axis of the vesicle, so the net orientation of the chromophores is close to vertical.

distribution of chromophores in the vesicle membrane means that the alignment of these chromophores with respect to the flow direction is not a significant issue. The elongation is essential for this; in a perfectly spherical vesicle with target membrane molecules randomly distributed throughout, there would be no net orientation of the chromophores. Vesicle elongation means that target molecules are more likely to be present in the membranes along the long axis of the vesicle than the short axis, leading to a net non-zero alignment of the molecules of concern (see Figure 1.13).

Since tank-treading is the most preferable vesicle behaviour for LD experiments, the phase diagram suggests that the main parameter of concern is Λ. In the experimental setup, the fluids inside and outside a vesicle (and, therefore, their viscosities) are the same, so we consider the case where the viscosity contrast $\lambda = 1$. Then,

$$\Lambda \approx 0.708 \times \sqrt{\Delta}\frac{\omega}{s}$$

As the aim is to keep the value of Λ low to ensure tank-treading occurs, this means that the excess area parameter, Δ, should be kept as small as possible, by not deforming the vesicle too much, and to ensure that the planar fluid flow is more extensional/elongational in character than rotational. Under extensional flows, if the vesicle is sufficiently deformable to give a small excess area (typically $\Delta < 0.1$), then it deforms into an ellipsoid aligned with the flow (with the vesicle long axis parallel to the "stretch" axis of the flow). If the

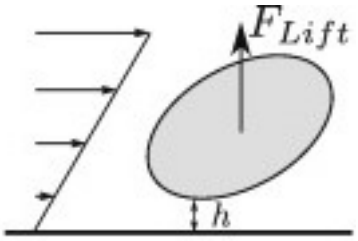


Figure 1.14: An illustration of the tilt of a vesicle away from a stationary boundary wall while in fluid shear flow. The vesicle is experiencing a lift force F_{lift}, whose magnitude is dependent upon the shear rate $\dot{\gamma}$. This diagram is reproduced with permission from the abstract figure of [2].

vesicle is more easily deformed and the excess area is large, then the membrane is likely to wrinkle. Furthermore, when the extensional strain rate is above a critical threshold, the vesicle stretch out as if it were a polymer uncoiling [2, 59, 106, 153, 168]. Therefore, in LD experiments, it is important that the lipid chosen to make up the vesicles allows for some shape deformation, but not too much.

This thesis considers the use of extensional flow and compares LD experimental results against those obtained using Couette shear flows. With regard to the shear flow as generated in a Couette flow cell used in LD experiments, there are a number of experimental and theoretical studies that describe the vesicle behaviour in these situations. GUVs (radius $10\,\mu$m) in simple shear flows (with $\dot{\gamma} \approx 1\,s^{-1}$) have been observed to elongate and tilt away from a stationary boundary wall (see Figure 1.14), with an inclination angle depending upon the reduced volume (calculated as $1 + (\Delta/4\pi))^{-3/2}$) and, to a lesser extent, the shear rate [57, 133]. Vesicles can also undergo tumbling behaviour in shear flows [1, 2, 11, 34, 57, 60, 167]. Other phenomena, including the lift force for vesicles moving along a stationary flow in shear fluid flow, are detailed in [2].

1.6 Microfluidic experiments with vesicles

In order to study vesicles in different fluid flows, we require a system for passing UV and visible-light radiation through a flow chamber that caters for the flow type required. Fortunately, there are such systems available that can be easily and cheaply manufactured and have been used to study vesicles in flow in the past [2].

Microelectro-mechanical systems were first developed in the 1980s for a variety of different purposes, including pressure sensors for automotive applications, chemical sensors, capacitors and vibration detectors [88, 144]. These devices also gave birth to the field of microfluidics, where devices could be used to study objects in fluid flows and control chemical reactions at the "micro" scale, smaller than what was previously possible (so-called "lab on a chip" technology). Earlier devices were based upon materials such as silicon and glass, which, while fabrication protocols were well known, made devices expensive to make. However the advent of polymer plastics have made the manufacture of microfluidic devices a cheaper process.

While a variety of different materials have been considered for use in making microfluidic devices, including the use of Shrinky Dinks® toys [33, 42, 95], one of the most commonly used materials is polydimethylsiloxane (PDMS) [87, 88, 144]. The first reported use of this material for microfluidic devices was in 1998 by Duffy et al. [31]. Sealed flow channels can be constructed by treating PDMS with oxygen plasma to create a hydrophilic surface that can be stuck on to a glass slide with heat treatment [31, 65, 145]. The polymer is also transparent to electromagnetic radiation with wavelengths in the range 240 nm to 1100 nm [87]. This allows us to use devices made from PDMS for linear dichroism experiments. However, there are some drawbacks to its use. For example, PDMS is hydrophobic and is able to absorb hydrophobic molecules from sample fluids that pass through it [92, 151]. This particular drawback could be a problem with studies of vesicle membrane proteins or membrane-bound molecules, as these molecules already have an affinity for the hydrophobic enviroment within the lipid bilayer.

An example of microfluidic devices applied to vesicle studies was performed by Kantsler and Steinberg [57], where the tank-treading motion of a vesicle was observed using a rectangular channel with 250 μm height and 150 μm width cast into PDMS by soft lithog-

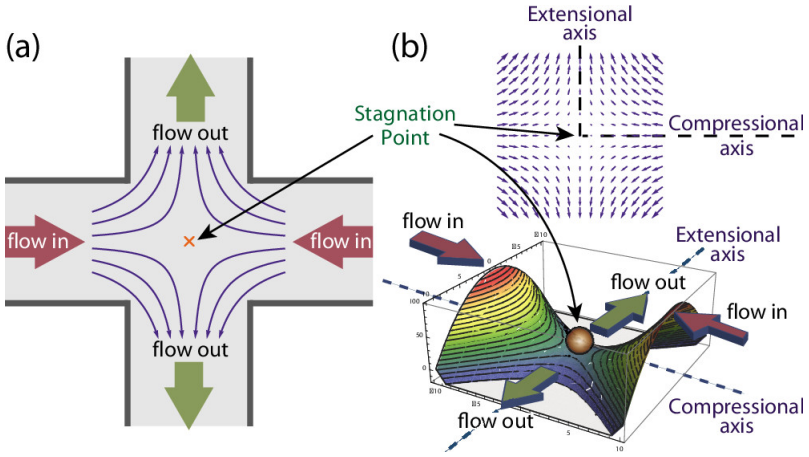


Figure 1.15: This diagram illustrates the cross-slot device used by Tanyeri et al. [147]. (a) A hydrodynamic trap is created by a planar extensional flow field at the junction of two perpendicular microchannels. (b) The velocity field (top) and the potential well (bottom) for a particle in the flow field at the microchannel junction. This diagram is reproduced with permission from Figure 1 of [147].

raphy. Extensional flows have also been studied with microfluidics. One such example is in work by Tanyeri et al. [147] who developed a particle trapping device using an extensional flow and a *cross-slot*. This is where the fluid enters a central chamber through two opposing channels on one axis (the compressional axis), and fluid leaves through two opposing channels on a perpendicular axis (the extensional axis), as illustrated in Figure 1.15). This device was used to trap objects of roughly 1 μm in diameter in the central stagnation point, using active flow controls to adjust the fluid flow rates exiting the cross-slot. Without such controls, such a particle would escape due to Brownian motion and the flow of fluid along the extensional axis [147, 148]. Cross-slot devices have also been used to study the uncoiling and extension of DNA molecules [135] and the behaviour of vesicles in extensional flows [59].

To study a range of different flows, including linear shear, extensional and rotational flows (as depicted in Figure 1.11), a more complicated setup, known as the *four-roll mill* is required. The original four-roll mill device was designed by Taylor [150] to investigate patterns formed by the emulsion of two fluids that do not mix. While Hudson et al. [49] developed a microfluidic analog of the four roll mill based on simulations, a more intuitive design was developed by Lee et al. [72] using fluid flows acting as the independently moving "mills" (see Figures 1.16 and 1.17). Different flow types in the central chamber can be generated by adjusting the flow rates Q_1 and Q_2 as shown in Figure 1.17. For example, an extensional flow can be generated by setting $Q_1/Q_2 = 1$, whereas a rotational flow can be generated by setting $Q_1/Q_2 = -1$. The device developed by Lee et al. [72]

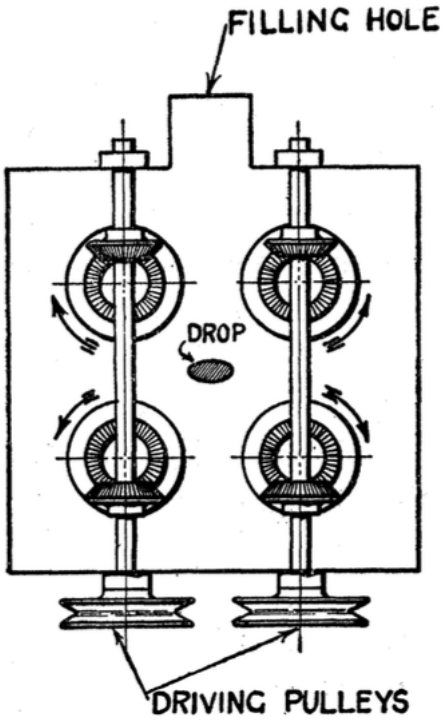


Figure 1.16: The four-roll mill device used by Taylor [150]. Four rotating mills controlled by driving pulleys cause the fluid drop in the centre to change shape. This diagram is reproduced with permission from Figure 1 of [150].

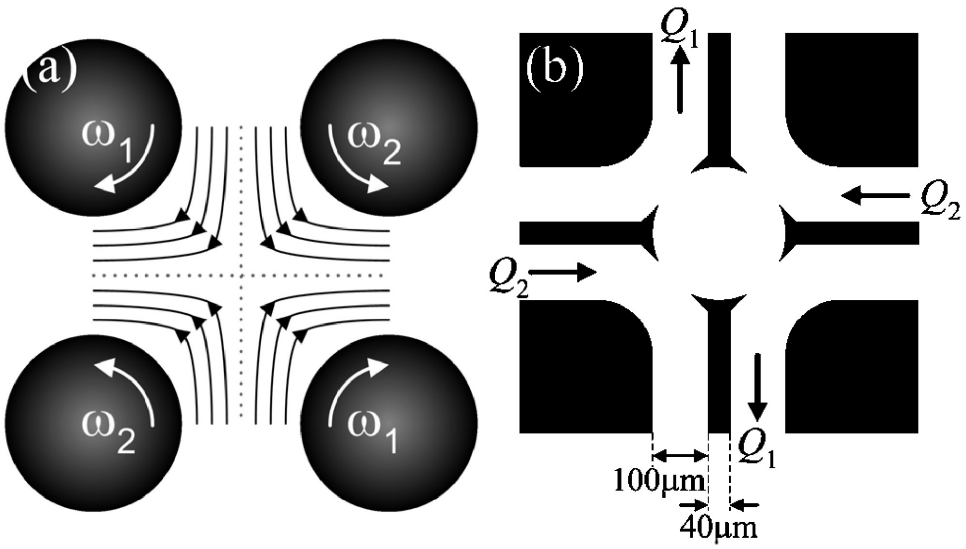


Figure 1.17: (a) A macroscale summary of the four-roll mill. (b) The microfluidic four-roll mill device used by Lee et al. [72]. All flow types can be generated by varying the angular velocity ratio ω_1/ω_2 or flow rate ratio Q_1/Q_2. These diagrams are reproduced with permission from Figure 1 of [72].

has been used to study the behaviour of vesicles in different planar fluid flows [26, 27, 76]. Of note is the work by Deschamps et al. [26], whose experimental results were shown to agree with the theoretical predictions established by Lebedev et al. [71].

It is again important to note that most of the studies considered here have used vesicles of at least 1 μm in diameter, studied using microscopy. Vesicles in studies using LD have diameters of the order of 100 nm. The goal of this thesis is to establish whether such devices can be used for vesicle membrane protein studies with linear dichroism spectroscopy and can achieve stronger LD signals through better alignment than possible with Couette flow. For extensional flow, the four-roll mill device would provide the extensional flows required for vesicle studies, but requires an extensive fluid pumping setup (Figure 1 in [76] provides an example of such a setup) that is beyond the scope of this project. Instead, an enhanced version of the cross-slot, as developed by Haward et al. [44] to optimize the area in the centre exposed to the greatest possible extensional strain, is used.

Before ending our discussion of microfluidic devices, it is worth noting that there is a linear dichroism spectroscopic study of a membrane protein in the literature that makes use of a flow-through, microfluidic device. Matsuo et al. [84] reported performing structural studies of α-lactalbumin, thioredoxin and β-lactoglobulin embedded in vesicles of roughly 100 nm in diameter using linear and circular dichroism. A microfluidic device consisting of a rectangluar window was fasioned out of two fused quartz plates, with a width of 2 mm, a length of 20 mm, and a 75 μm path length for light to pass through the sample. The vesicle sample was pumped through the cell as a pressure-driven (or Poiseuille[5]) flow. With this experiment in mind, a similar device constructed using PDMS is used in this investigation.

[5] Named after French physicist, Jean Léonard Marie Poiseuille

1.7 Thesis outline

The rest of this thesis is roughly split into two parts. The first part consists of work pertaining to the simulation of polymers in fluid flows using a FENE-Fraenkel spring force law. Chapter 2 focusses on this force law and contains analytically derived properties that are used to test the simulation under no-flow conditions. It also outlines the algorithm used to run bead-spring simulations. Chapter 3 details the results from simulations, including comparisons with data reported by Hsieh et al. [48] to assess the performance of our code, and a test of the effect of hydrodynamic interactions on the alignment parameter value obtained.

The second part of the thesis focuses on experimental work with microfluidic devices, DNA and vesicles. Linear dichroism signals with DNA in microfluidic devices under the influence of pressure-driven, Couette shear and extensional flows are explored in Chapter 4. Vesicles are studied in the same manner in Chapter 5 with the use of a membrane probe molecule to act as a membrane protein mimic.

Chapter 2

Using FENE-Fraenkel springs in polymer simulations

Bead-rod and bead-spring chain simulations have been used to study the behaviour of polymer molecules, such as DNA, in fluid flows. Chapter 1 contains examples of different spring force laws as used in bead-spring simulations. This chapter focuses on the FENE-Fraenkel force law, using both the FENE and Fraenkel spring force laws as a motivation for construction. Equilibrium properties of the FENE-Fraenkel spring are calculated, and are used in the next chapter to assess the performance of a simulation programme. The algorithm used to simulate bead-spring chains in flow with the FENE-Fraenkel spring is based on a code developed by [117] for FENE springs; the details of this algorithm are summarised in this chapter; intricate programme code details can be found in the appendices.

2.1 Spring Force Law Formulation

2.1.1 FENE springs

In the bead-spring simulation literature, a finitely extensible nonlinear elastic (FENE) spring is used to model a situation where a spring cannot be extended beyond a certain length, and where the tension force increases non-linearly as the spring length reaches this maximum permissible length.

Expressing this mathematically, let **Q** the spring vector; this is a vector pointing in the same direction as the spring with length equal to the spring length, as illustrated in Figure 2.1.

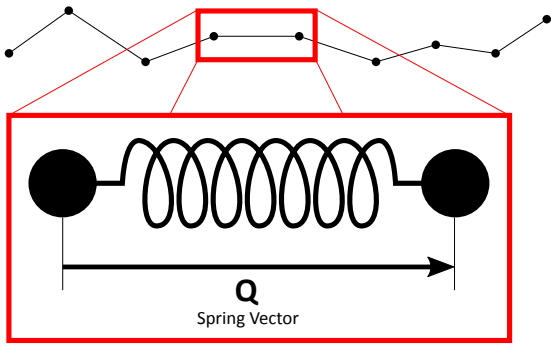


Figure 2.1: An illustration of a spring vector of a spring in a bead-spring chain.

The spring tension force, **F**, exerted by a FENE spring as a function of the spring vector **Q** is

$$\mathbf{F} = \frac{H\mathbf{Q}}{1 - \left(\frac{Q}{Q_{\max}}\right)^2} \tag{2.1.1}$$

where

- $H > 0$ is the spring constant.
- $Q = |\mathbf{Q}|$ is the length of the spring.
- $Q_{\max} > 0$ is the maximum length of the spring.
- $0 \leq Q < Q_{\max}$ describes the range of permissible values of the spring length.

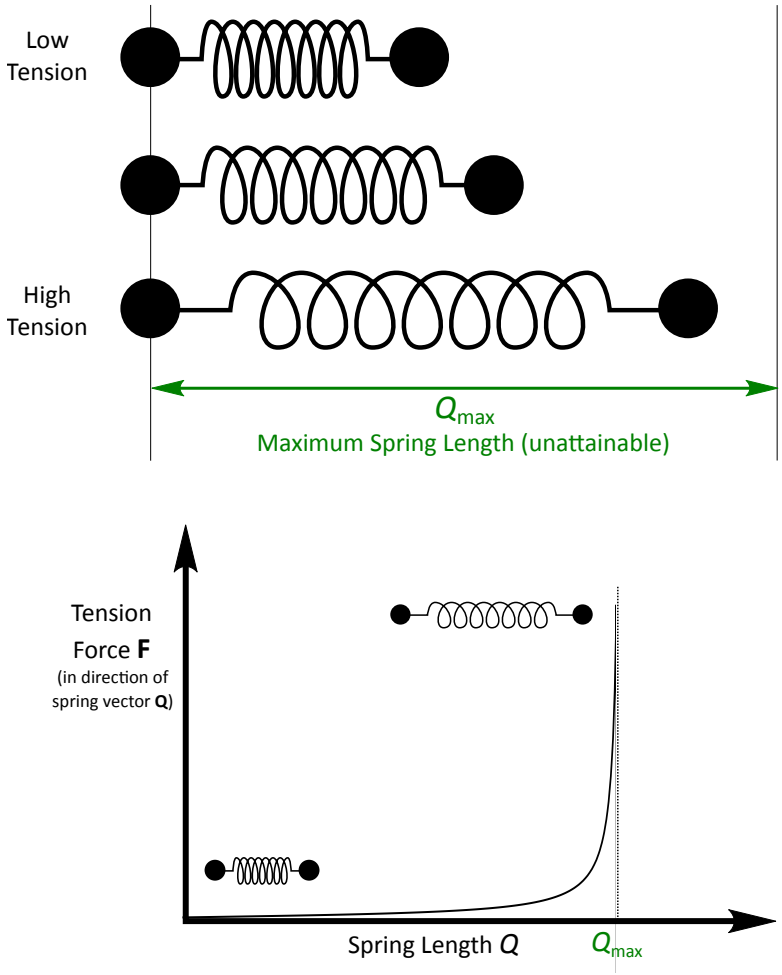


Figure 2.2: An illustration of the relationship between the length and the spring tension of a FENE spring. The spring length can vary between zero inclusive and $Q_{\max}$ exclusive. The tension force increases in magnitude significantly as the spring length gets close to $Q_{\max}$.

The form of the denominator in Equation (2.1.1) means that, for values of Q close to $Q_{\max}$, the magnitude of the spring force should rapidly increase and "blow up" to infinity when the spring has reached its maximum permissible length (see Figure 2.2). Furthermore, for values of Q close to 0, the force is almost linearly proportional to the spring length; the behaviour of the spring is said to be "Hookean" for small spring lengths [14].

2.1.2 Fraenkel springs

Another type of spring used in the bead-spring simulation literature is the Fraenkel spring. Whereas in a FENE spring, the force is a function of the total spring length, a Fraenkel spring has a pre-deteremined natural length and the tension is directly proportional to the *extension or compression* of the spring from this length (see Figure 2.3).

The force, **F**, exerted by a Fraenkel spring as a function of the spring vector **Q** is

$$\mathbf{F} = H\,(Q - Q_0)\,\frac{\mathbf{Q}}{Q} \tag{2.1.2}$$

where

- $H > 0$ is the spring constant.
- $Q = |\mathbf{Q}|$ is the length of the spring.
- $Q_0 \in [0, \infty)$ is the natural length of the spring.
- $0 < Q < \infty$ describes the range of permissible values of the spring length.

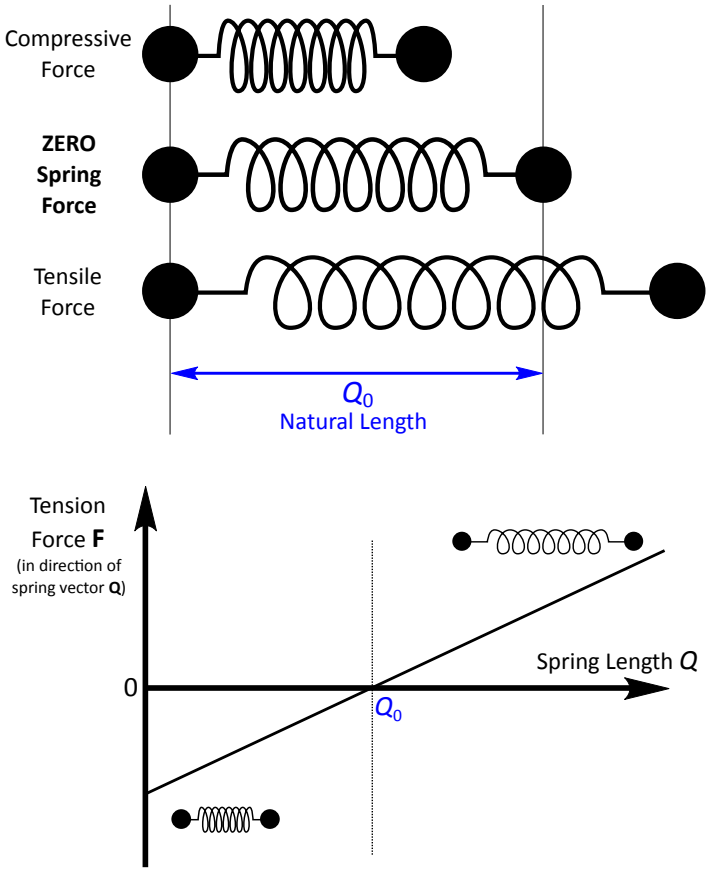


Figure 2.3: An illustration of the relationship between the length and the spring tension of a Fraenkel spring. The spring tension/compression force is linearly proportional to the extension/compression from the natural spring length Q_0.

The $(Q - Q_0)$ term in Equation (2.1.2) ensures that the force is proportional to the spring extension or compression from the natural length, with spring extension leading to a positive (tensile) force, and spring compression leading to a negative ("pushing-out") force. The fraction term, $\mathbf{Q}/Q$, provides for a normalised vector in the direction of the spring.

2.1.3 FENE-Fraenkel stiff spring

Taking the element of tension being proportional to the deviation from a natural length from the Fraenkel spring and combining this with the nonlinear proportionality between spring length & tension and the maximum extensibility/compresibility of the FENE spring gives a formulation known as a FENE-Fraenkel spring. It is known as a "stiff spring" setup because it behaves similarly to a rod, but with the allowance for a small amount of compression or stretching. This hybrid of the two behaviours is what, arguably, makes it a better spring force law to use in a bead-spring model when studying polymers such as DNA.

The expression for the spring tension force in this instance is

$$\mathbf{F} = \frac{H\,(Q - Q_0)}{1 - \left(\frac{Q-Q_0}{\delta}\right)^2} \frac{\mathbf{Q}}{Q} \tag{2.1.3}$$

where

- $H > 0$ is the spring constant.
- $Q = |\mathbf{Q}|$ is the length of the spring.
- $Q_0 \in [0, \infty)$ is the natural length of the spring.
- $\delta \in (0, Q_0)$ is the maximum stretch/compression from the natural length.
- $Q_0 - \delta < Q < Q_0 + \delta$ describes the range of permissible values of the spring length. This is equivalent to describing the range of permissible deviations from the natural length as: $-\delta < Q - Q_0 < \delta$.

The $\mathbf{Q}/Q$ term provides for a normalised vector in the direction of the spring. When the spring length Q is close to Q_0, the tension is roughly linearly proportional to extension or compression, as measured from the natural length Q_0. That is, when $(Q - Q_0)/\delta << 1$,

$$\mathbf{F} \approx H\,(Q - Q_0)\,\frac{\mathbf{Q}}{Q}. \tag{2.1.4}$$

The $(Q - Q_0)$ term provides a sign to the force, with spring extension leading to a positive (tensile) force pulling the beads closer together, and spring compression leading to a force pushing the beads apart. If Q increases towards the maximum value of $Q_0 + \delta$, observe

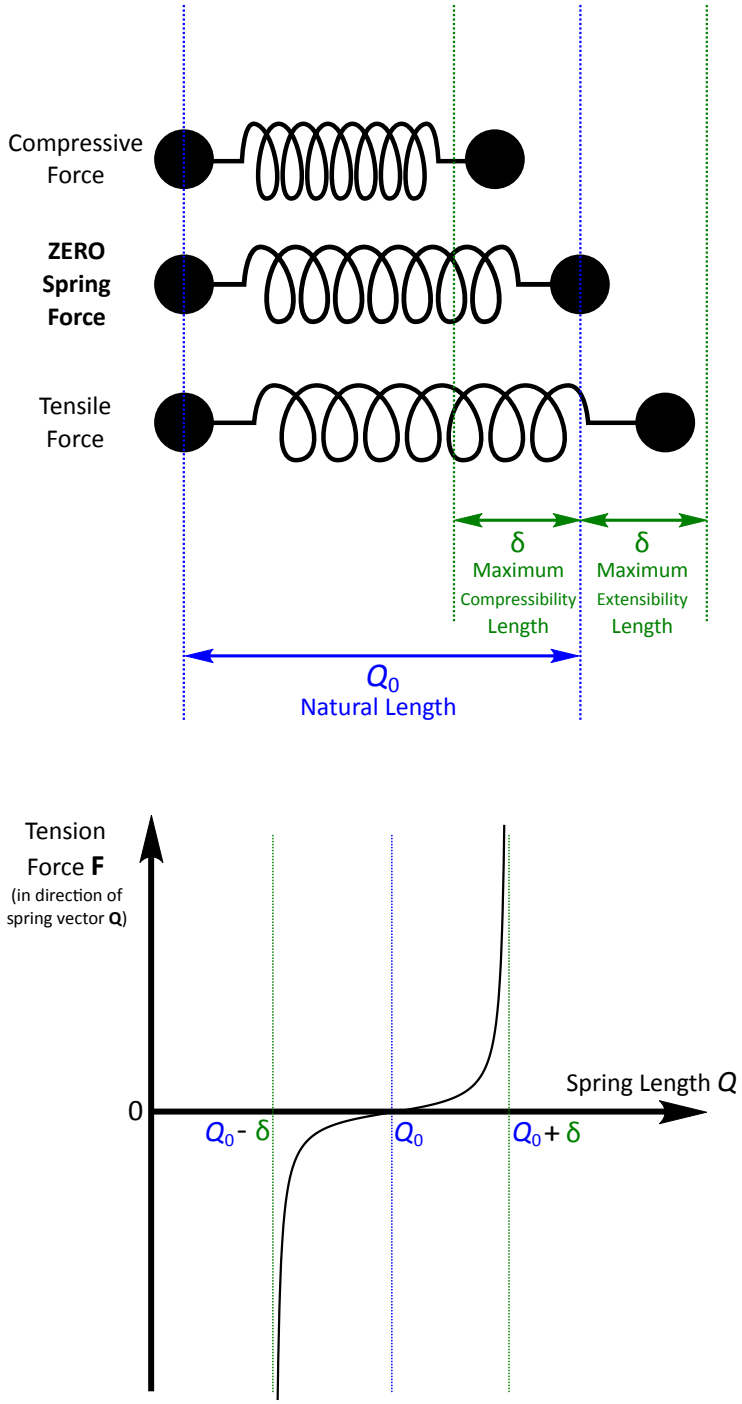


Figure 2.4: An illustration of the relationship between the length and the spring tension of a FENE-Fraenkel spring. The tension/compression force is zero when the spring is at its natural length Q_0, and increases in magnitude with stretching/compression from this length, approaching infinity when the spring length gets close to $Q_0 - \delta$ or $Q_0 + \delta$.

that,

$$\lim_{Q \nearrow Q_0+\delta} \left(1 - \left(\frac{Q - Q_0}{\delta}\right)^2\right) = 1 - \left(\frac{Q_0 + \delta - Q_0}{\delta}\right)^2 = 1 - \left(\frac{\delta}{\delta}\right)^2 = 0.$$

Thus the magnitude of the spring force, F, blows up to $+\infty$ in the direction of the spring; that is, the spring is pulling the beads together with an infinitely sized force. A similar situation can be derived for beads being pushed apart with an infinitely sized spring force when Q approaches $Q_0 - \delta$. This is illustrated in Figure 2.4.

2.2 Non-dimensionalised forms of the spring force laws

In order to enable comparison of data generated using the equations in Section 2.1 and data from other sources, we need to non-dimensionalise the spring force law equations. For this section, asterisks (*) are used to indicate non-dimensionalised values. The following scales are used to non-dimensionalise variables within the equations:

- $\sqrt{\frac{k_b T}{H}}$ is the length scale.
- $\sqrt{k_b T H}$ is the force scale.

where k_B is Boltzmann's constant, T is the absolute temperature and H is the spring constant. This is to maintain consistency with [108] and [53]. So, in particular, we have:

$$Q_0 = \sqrt{\frac{k_b T}{H}} Q_0^* \qquad \text{and} \qquad \mathbf{F} = \mathbf{F}^* \sqrt{k_b T H}.$$

2.2.1 FENE force law

The non-dimensionalised form of the FENE force law is

$$\mathbf{F}^* = \frac{\mathbf{Q}^*}{1 - \left(\frac{Q^*}{\sqrt{b}}\right)^2}, \tag{2.2.1}$$

where

- $Q^* = |\mathbf{Q}^*|$ is the non-dimensional length of the spring.
- $b = \frac{(Q_{\max})^2}{\frac{k_b T}{H}}$ is the squared non-dimensional maximum stretch/compression.
- $0 \leq Q^* < \sqrt{b}$ describes the range of permissible values of the spring length.

2.2.2 Fraenkel force law

The non-dimensionalised form of the Fraenkel force law is

$$\mathbf{F}^* = \left(1 - \frac{Q_0^*}{Q^*}\right)\mathbf{Q}^*, \tag{2.2.2}$$

where

- $Q^* = |\mathbf{Q}^*|$ is the length of the spring.
- $Q_0^* > 0$ is the natural length of the spring.
- $0 < Q^* < \infty$ describes the range of permissible values of the spring length.

2.2.3 FENE-Fraenkel force law

The non-dimensionalised form of the FENE-Fraenkel force law.

$$\mathbf{F}^* = \frac{\left(1 - \frac{Q_0^*}{Q^*}\right)}{1 - \left(\frac{Q^* - Q_0^*}{\delta^*}\right)^2}\mathbf{Q}^*, \tag{2.2.3}$$

where

- $Q^* = |\mathbf{Q}^*|$ is the length of the spring.
- $Q_0^* > 0$ is the natural length of the spring.
- $\delta^* = \dfrac{\delta}{\sqrt{\frac{k_b T}{H}}} \in (0, Q_0)$ is the maximum stretch/compression from the natural length.
- $0 < Q_0^* - \delta^* < Q^* < Q_0^* + \delta^*$ describes the range of permissible values of the spring length.

For the rest of this chapter, unless stated otherwise, the variables and force laws are refered to in their non-dimensionalised form listed above and appear without the asterisk (*) marks.

2.3 Regenerating the FENE and Fraenkel force laws

In work by Jain et al. [53] on the use of FENE springs, the parameter $\sqrt{b}$ is used to describe the maximum non-dimensionalised length of the spring. Thus,

$$b = \frac{Q_{\max}^2}{(\text{Length scale})^2},$$

where $Q_{\max}$ is the maximum extensibility in the FENE spring (as described earlier). Comparing the non-dimensional forms of the FENE and FENE-Fraenkel force laws, one can see that the extensibility parameter δ in the FENE-Fraenkel model is analogous to the $\sqrt{b}$ parameter in the FENE model.

2.3.1 Obtaining the FENE law from FENE-Fr. law

The FENE-Fraenkel model requires $Q_0 - \delta < Q < Q_0 + \delta$, but the FENE model requires $0 < Q < \sqrt{b}$. So consider the limiting situation as $Q_0 \to 0$ and $\delta \to \sqrt{b}$.

$$\lim_{Q_0 \to 0,\, \delta \to \sqrt{b}} \mathbf{F} = \lim_{Q_0 \to 0,\, \delta \to \sqrt{b}} \left(\frac{\left(1 - \frac{Q_0}{Q}\right)}{1 - \left(\frac{Q - Q_0}{\delta}\right)^2} \mathbf{Q} \right) = \frac{\left(1 - \frac{0}{Q}\right)}{1 - \left(\frac{Q}{\sqrt{b}}\right)^2} \mathbf{Q} = \frac{\mathbf{Q}}{1 - \left(\frac{Q}{\sqrt{b}}\right)^2}$$

This is analogous to the FENE force law given by Equation (2.2.1), where the maximum stretch length is now $\sqrt{b}$. However, care needs to be taken with the range of lengths that the spring can attain. In the FENE-Fraenkel model, $Q_0 - \delta < Q < Q_0 + \delta$. But with $Q_0 = 0$, this gives: $-\sqrt{b} < Q < \sqrt{b}$; this is a wider range than that permitted for the FENE model ($0 < Q < \sqrt{b}$).

2.3.2 Obtaining the Fraenkel law from FENE-Fr. law

In contrast, regenerating the Fraenkel setup requires very little effort. The Fraenkel setup uses a spring with a non-zero natural length Q_0. Furthermore, the only limit to the spring's

length is that it cannot reach zero. If we take the limit as $\delta \to \infty$. Then,

$$\lim_{\delta\to\infty} \mathbf{F} = \lim_{\delta\to\infty} \frac{\left(1 - \frac{Q_0}{Q}\right)}{1 - \left(\frac{Q-Q_0}{\delta}\right)^2}\mathbf{Q} = \frac{\left(1 - \frac{Q_0}{Q}\right)}{1 - \lim_{\delta\to\infty}\left(\frac{Q-Q_0}{\delta}\right)^2}\mathbf{Q} = \frac{\left(1 - \frac{Q_0}{Q}\right)}{1}\mathbf{Q} = \left(1 - \frac{Q_0}{Q}\right)\mathbf{Q}$$

The last formula is consistent with the non-dimensionalised form of the Fraenkel setup in Equation (2.2.2).

2.4 Zero-shear rate equilibrium properties

To determine if simulations are outputting sensible values, it is preferable to be able to compare simulation data with analytically derived results. To this end, various equilibrium properties of a bead-spring chain made up of identical FENE-Fraenkel springs are derived in this section.

2.4.1 Calculating equilibrium properties

For the beginning of this subsection, dimensionalised parameters and variables are used. Then non-dimensionalised parameters and variables for the FENE-Fraenkel force law as given in Equation (2.2.3) *are considered; these are asterisked (*).*

Consider a chain with N beads connected by $N-1$ springs, with spring vectors represented by $\mathbf{Q}^{N-1}$. For a property B which only depends on the spring vector coordinates of a bead-spring chain (not on the position of the chain in space) and in a situation where the potential energy of the chain in a system depends only on the chain configuration, the equilbrium value for this property, $\langle B \rangle$, is given by

$$\langle B \rangle = \int B\, \psi_{eq}\left(\mathbf{Q}^{N-1}\right) \mathrm{d}\mathbf{Q}^{N-1} \tag{2.4.1}$$

where $\psi_{eq}\left(\mathbf{Q}^{N-1}\right)$ is the distribution function describing the probability that a chain has the particular configuration with spring vectors $\mathbf{Q}^{N-1}$ and ignoring momenta of beads in the chain and the centre of mass of the chain [14, Equation 12.4-5, §12.4].

Furthermore, if it is assumed that, for all beads in the chain, there is no restriction on

the angle between the two springs connected to the same bead, then the chain is said to be *freely-jointed*. The associated distribution function, $\psi_{eq}\left(\mathbf{Q}^{N-1}\right)$, for a freely jointed bead-spring chain with N beads is

$$\psi_{eq}\left(\mathbf{Q}^{N-1}\right) = \frac{\exp\left(-\phi\left(\mathbf{Q}^{N-1}\right)/k_BT\right)}{\int \exp\left(-\phi\left(\mathbf{Q}^{N-1}\right)/k_BT\right)\,\mathrm{d}\mathbf{Q}^{N-1}}, \tag{2.4.2}$$

where k_B and T are Boltzmann's constant and the absolute temperature and $\phi(\mathbf{Q}^{N-1})$ is the spring potential function [14, Equation 12.3-7, §12.3]. ϕ is related to the spring force law $\mathbf{F}$ by [14, §12.2]:

$$\mathbf{F} = \frac{\partial\phi}{\partial\mathbf{Q}} \tag{2.4.3}$$

Before using these expressions with the spring-force laws described above, it is important to check what effect non-dimensionalisation of variables has on these functions. Observe that with the length and force scale factors previously discussed in Section 2.2, the corresponding energy scale factor is k_BT. So with

- $\mathbf{F} = \mathbf{F}^* \times \sqrt{k_BTH}$
- $\mathbf{Q} = \mathbf{Q}^* \times \sqrt{\frac{k_BT}{H}}$
- $\phi = \phi^* \times k_BT$

we get

$$\mathbf{F}^* = \frac{\mathbf{F}}{\sqrt{k_BTH}} = \frac{1}{\sqrt{k_BTH}}\frac{\partial\phi}{\partial\mathbf{Q}} = \frac{1}{\sqrt{k_BTH}}\frac{k_BT}{\sqrt{\frac{k_BT}{H}}}\frac{\partial\phi^*}{\partial\mathbf{Q}^*} = \frac{\partial\phi^*}{\partial\mathbf{Q}^*}.$$

Furthermore,

$$\psi_{eq}\left(\mathbf{Q}^*\right) = \frac{\exp\left(-\phi^*\left(\mathbf{Q}^*\right)\right)}{\int \exp\left(-\phi^*\right)\,\mathrm{d}\mathbf{Q}^*}. \tag{2.4.4}$$

For the rest of the section, it will be assumed that all variables are non-dimensionalised and the asterisks (*) are omitted.

2.4.2 Potential function ϕ for a dumbbell

In this model, the behaviour of springs within a bead-spring chain affect each other only in the position of the beads that they are connected to. That is, the spring force for any individual spring in a chain is dependent only on the length of that spring, and not of the length of other springs. With this in mind, the potential function is calculated below for a dumbbell setup — a chain made up of one spring and two beads.

FENE Force Law

Taking the non-dimensionalised FENE force law (Equation (2.2.1)) as our starting point,

$$\mathbf{F} = \frac{\mathbf{Q}}{1 - \left(\frac{Q}{\sqrt{b}}\right)^2} = \frac{\mathbf{Q}}{1 - \left(\frac{\mathbf{Q}\cdot\mathbf{Q}}{b}\right)} = \frac{\mathbf{Q}}{1 - \left(\frac{\mathbf{Q}^2}{b}\right)}.$$

Integrating this force expression with respect to the spring length gives the potential function:

$$\begin{aligned}\phi &= \int \mathbf{F}\,\mathrm{d}\mathbf{Q} = \int \frac{\mathbf{Q}}{1 - \left(\frac{\mathbf{Q}^2}{b}\right)}\,\mathrm{d}\mathbf{Q} = \frac{-b}{2}\int \frac{\mathrm{d}}{\mathrm{d}\mathbf{Q}}\left(\ln\left(1 - \frac{\mathbf{Q}^2}{b}\right)\right)\mathrm{d}\mathbf{Q} \\ &= \frac{-b}{2}\ln\left(1 - \frac{\mathbf{Q}^2}{b}\right) = -\frac{b}{2}\ln\left(1 - \left(\frac{Q}{\sqrt{b}}\right)^2\right) \\ &= -\frac{b}{2}\ln\left(1 - q^2\right)\end{aligned} \tag{2.4.5}$$

where $q = \dfrac{Q}{\sqrt{b}}$. This is consistent with the expression in Equation (19) of [108].

FENE-Fraenkel Force Law

Rearranging the RHS of Equation (2.2.3) gives:

$$\begin{aligned}\mathbf{F} &= \frac{(\mathbf{Q} - \mathbf{Q}_0)}{1 - \left(\frac{Q - Q_0}{\delta}\right)^2} = \delta\left(\frac{\mathbf{Q} - \mathbf{Q}_0}{\delta}\right)\left(1 - \left(\frac{(\mathbf{Q} - \mathbf{Q}_0)^2}{\delta^2}\right)\right)^{-1} \\ &= -\frac{\delta^2}{2} \times \frac{-2}{\delta}\left(\frac{\mathbf{Q} - \mathbf{Q}_0}{\delta}\right) \times \left(1 - \left(\frac{\mathbf{Q} - \mathbf{Q}_0}{\delta}\right)^2\right)^{-1} \\ &= -\frac{\delta^2}{2}\frac{\partial}{\partial \mathbf{Q}}\left(\ln\left(1 - \left(\frac{\mathbf{Q} - \mathbf{Q}_0}{\delta}\right)^2\right)\right)\end{aligned}$$

Thus,

$$\begin{aligned}\phi &= \frac{-\delta^2}{2}\ln\left(1-\left(\frac{\mathbf{Q}-\mathbf{Q}_0}{\delta}\right)^2\right)\\ &= \frac{-\delta^2}{2}\ln\left(1-\frac{|\mathbf{Q}-\mathbf{Q}_0|^2}{\delta^2}\right)\\ &= \frac{-\delta^2}{2}\ln\left(1-\frac{(Q-Q_0)^2}{\delta^2}\right)\\ &= \frac{-\delta^2}{2}\ln\left(1-(q-q_0)^2\right)\end{aligned} \tag{2.4.6}$$

where $q = Q/\delta$ and $q_0 = Q_0/\delta$ and the directions of $\mathbf{Q}$ and $\mathbf{Q}_0$ are identical (hence $|\mathbf{Q}-\mathbf{Q}_0|^2 = (Q-Q_0)^2$.)

2.4.3 Equilibrium properties of the length of a FENE-Fraenkel spring

The integrals in Equations (2.4.1) and (2.4.2) are performed over cartesian coordinates. If the properties being considered are related to spring lengths, it will be easier to switch to spherical-polar coordinates [14, Equation 12.5-3, §12.5]. In the case of a single spring in a dumbbell:

$$\begin{aligned}\psi_{eq}(\mathbf{Q}) = \psi_{eq}(Q,\theta,\varphi) &= \psi_{eq}\left(Q_x,Q_y,Q_z\right)\frac{\partial\left(Q_x,Q_y,Q_z\right)}{\partial(Q,\theta,\varphi)}\\ &= \psi_{eq}\left(Q_x,Q_y,Q_z\right)Q^2\sin\theta,\end{aligned} \tag{2.4.7}$$

where $\left(Q_x,Q_y,Q_z\right)$ are the cartesian coordinates and (Q,θ,φ) are the spherical-polar coordinates representing spring vector $\mathbf{Q}$.

To demonstrate the effect this change of coordinates has, consider the equilibrium value of the squared length of a bead-spring dumbbell. This is known as the *second moment*.

$$\begin{aligned}\langle\mathbf{Q}^2\rangle &= \int(\mathbf{Q}\cdot\mathbf{Q})\,\psi_{eq}(\mathbf{Q})\,\mathrm{d}\mathbf{Q} = \iiint Q^2\,\psi_{eq}(\mathbf{Q})\,\mathrm{d}Q_x\,\mathrm{d}Q_y\,\mathrm{d}Q_z\\ &= \iiint Q^2\,\psi_{eq}(\mathbf{Q})\,Q^2\sin\theta\,\mathrm{d}Q\,\mathrm{d}\theta\,\mathrm{d}\varphi = \int Q^4\,\psi_{eq}(\mathbf{Q})\sin\theta\,\mathrm{d}Q\,\mathrm{d}\theta\,\mathrm{d}\varphi\\ &= \frac{\iiint Q^4\sin\theta\exp\left(-\phi(Q)\right)\,\mathrm{d}Q\,\mathrm{d}\theta\,\mathrm{d}\varphi}{\iiint Q^2\sin\theta\exp\left(-\phi(Q)\right)\,\mathrm{d}Q\,\mathrm{d}\theta\,\mathrm{d}\varphi}\end{aligned} \tag{2.4.8}$$

Since there are no terms in the numerator and denominator dependent on φ, integration with respect to this variable results in the same constant on both top and bottom, thus

cancelling each other out. Furthermore, integration with respect to θ is done in the range $[0, \pi]$. As the only term depending on θ in both numerator and denominator is $\sin\theta$, integration with respect to this variable also results in constants which cancel each other out. Therefore:

$$\langle \mathbf{Q}^2 \rangle = \frac{\iiint Q^4 \sin\theta \exp\left(-\phi(Q)\right) \, \mathrm{d}Q \, \mathrm{d}\theta \, \mathrm{d}\varphi}{\iiint Q^2 \sin\theta \exp\left(-\phi(Q)\right) \, \mathrm{d}Q \, \mathrm{d}\theta \, \mathrm{d}\varphi} = \frac{\int Q^4 \exp\left(-\phi(Q)\right) \, \mathrm{d}Q}{\int Q^2 \exp\left(-\phi(Q)\right) \, \mathrm{d}Q}. \tag{2.4.9}$$

The calculations below involve integrals of the form that appear in the right-hand side of Equation (2.4.9). Beta functions will be used to process these.

A *Beta function* is a function $\mathrm{B}(x, y)$ on two complex numbers, $x, y \in \mathbb{C}$ with $\mathrm{Re}(x), \mathrm{Re}(y) > 0$, given by the following integral:

$$\mathrm{B}(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1} \, \mathrm{d}t. \tag{2.4.10}$$

The following result is used in the calculations of the second and fourth moments: For any $x, y \in \mathbb{C}$ with $\mathrm{Re}(x), \mathrm{Re}(y) > 0$,

$$\mathrm{B}(x+1, y) = \mathrm{B}(x, y)\frac{x}{x+y}. \tag{2.4.11}$$

A proof of this result is given in Appendix A.

Second moment

As explained above, the equilibrium average of the second moment is given by:

$$\langle \mathbf{Q}^2 \rangle = \langle Q^2 \rangle = \frac{\int_{Q_0-\delta}^{Q_0+\delta} Q^4 e^{-\phi} \, \mathrm{d}Q}{\int_{Q_0-\delta}^{Q_0+\delta} Q^2 e^{-\phi} \, \mathrm{d}Q} \tag{2.4.12}$$

To simplify calculations below, let q and q_0 be defined by:

$$q = \frac{Q}{\delta} \quad \text{with} \quad \frac{\mathrm{d}Q}{\mathrm{d}q} = \delta \quad \text{and} \quad q_0 = \frac{Q_0}{\delta}.$$

From Subsection 2.4.2 above,

$$\begin{aligned}\exp(-\phi) &= \exp\left(\frac{\delta^2}{2}\ln\left(1-(q-q_0)^2\right)\right)\\ &= \exp\left(\ln\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\right)\\ &= \left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\end{aligned}$$

Therefore:

$$\begin{aligned}\langle Q^2\rangle &= \frac{\int_{q_0-1}^{q_0+1}\delta^4 q^4\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\frac{\mathrm{d}Q}{\mathrm{d}q}\,\mathrm{d}q}{\int_{q_0-1}^{q_0+1}\delta^2 q^2\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\frac{\mathrm{d}Q}{\mathrm{d}q}\,\mathrm{d}q}\\ &= \delta^2\frac{\int_{q_0-1}^{q_0+1}q^4\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\,\mathrm{d}q}{\int_{q_0-1}^{q_0+1}q^2\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\,\mathrm{d}q} = \delta^2\frac{\boxed{\text{A}}}{\boxed{\text{B}}}\end{aligned}$$

Integrals $\boxed{\text{A}}$ and $\boxed{\text{B}}$ are considered in turn below.

Spliting the integration region of $\boxed{\text{A}}$ at q_0 gives two separate integrals to deal with.

$$\begin{aligned}\boxed{\text{A}} &= \int_{q_0-1}^{q_0+1}q^4\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\,\mathrm{d}q\\ &= \int_{q_0}^{q_0+1}q^4\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\,\mathrm{d}q + \int_{q_0-1}^{q_0}q^4\left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\,\mathrm{d}q\\ &= \boxed{\text{A1}} + \boxed{\text{A2}}\end{aligned}$$

Integral $\boxed{\text{A1}}$ is solved using the following substitution.

$$t = (q-q_0)^2 \quad \text{with} \quad \sqrt{t} = q - q_0 \quad \text{and} \quad \frac{\mathrm{d}q}{\mathrm{d}t} = \frac{1}{2\sqrt{t}}.$$

Then, applying the beta function formula Equation (2.4.10) gives

$$\begin{aligned}
\boxed{\text{A1}} &= \int_0^1 \frac{\left(\sqrt{t}+q_0\right)^4}{2\sqrt{t}}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&= \int_0^1 \frac{\left(t^2+4q_0t^{3/2}+6q_0^2t+4q_0^3t^{1/2}+q_0^4\right)}{2\sqrt{t}}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&= \frac{1}{2}\int_0^1 t^{3/2}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t + 2q_0\int_0^1 t^1(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t + 3q_0^2\int_0^1 t^{1/2}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&\quad + 2q_0^3\int_0^1 t^0(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t + \frac{q_0^4}{2}\int_0^1 t^{-1/2}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&= \frac{1}{2}\mathrm{B}\left[\frac{5}{2},\frac{\delta^2+2}{2}\right] + 2q_0\mathrm{B}\left[2,\frac{\delta^2+2}{2}\right] + 3q_0^2\mathrm{B}\left[\frac{3}{2},\frac{\delta^2+2}{2}\right] \\
&\quad + 2q_0^3\mathrm{B}\left[1,\frac{\delta^2+2}{2}\right] + \frac{q_0^4}{2}\mathrm{B}\left[\frac{1}{2},\frac{\delta^2+2}{2}\right]
\end{aligned}$$

For $\boxed{\text{A2}}$, a slightly different substitution is used.

$$t = (q-q_0)^2 \quad \text{but} \quad -\sqrt{t} = q-q_0 \quad \text{and} \quad \frac{\mathrm{d}q}{\mathrm{d}t} = \frac{-1}{2\sqrt{t}}.$$

The addition of minus signs is required as $t \geq 0$, but $q \in [q_0-1, q_0]$.

Using this substitution and applying the beta function expression in Equation (2.4.10),

$$\begin{aligned}
\boxed{\text{A2}} &= \int_1^0 \frac{\left(\sqrt{t}+q_0\right)^4}{-2\sqrt{t}}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t = \int_0^1 \frac{\left(\sqrt{t}+q_0\right)^4}{2\sqrt{t}}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&= \int_0^1 \frac{\left(t^2-4q_0t^{3/2}+6q_0^2t-4q_0^3t^{1/2}+q_0^4\right)}{2\sqrt{t}}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&= \frac{1}{2}\int_0^1 t^{3/2}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t - 2q_0\int_0^1 t^1(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t + 3q_0^2\int_0^1 t^{1/2}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&\quad - 2q_0^3\int_0^1 t^0(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t + \frac{q_0^4}{2}\int_0^1 t^{-1/2}(1-t)^{\frac{\delta^2}{2}}\,\mathrm{d}t \\
&= \frac{1}{2}\mathrm{B}\left[\frac{5}{2},\frac{\delta^2+2}{2}\right] - 2q_0\mathrm{B}\left[2,\frac{\delta^2+2}{2}\right] + 3q_0^2\mathrm{B}\left[\frac{3}{2},\frac{\delta^2+2}{2}\right] \\
&\quad - 2q_0^3\mathrm{B}\left[1,\frac{\delta^2+2}{2}\right] + \frac{q_0^4}{2}\mathrm{B}\left[\frac{1}{2},\frac{\delta^2+2}{2}\right]
\end{aligned}$$

Therefore

$$\begin{aligned}
\boxed{\mathrm{A}} &= \boxed{\mathrm{A1}} + \boxed{\mathrm{A2}} \\
&= \frac{1}{2}\mathrm{B}\left[\frac{5}{2}, \frac{\delta^2+2}{2}\right] + 2q_0\mathrm{B}\left[2, \frac{\delta^2+2}{2}\right] + 3q_0^2\mathrm{B}\left[\frac{3}{2}, \frac{\delta^2+2}{2}\right] \\
&\quad + 2q_0^3\mathrm{B}\left[1, \frac{\delta^2+2}{2}\right] + \frac{q_0^4}{2}\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right] \\
&\quad + \frac{1}{2}\mathrm{B}\left[\frac{5}{2}, \frac{\delta^2+2}{2}\right] - 2q_0\mathrm{B}\left[2, \frac{\delta^2+2}{2}\right] + 3q_0^2\mathrm{B}\left[\frac{3}{2}, \frac{\delta^2+2}{2}\right] \\
&\quad - 2q_0^3\mathrm{B}\left[1, \frac{\delta^2+2}{2}\right] + \frac{q_0^4}{2}\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right] \\
&= \mathrm{B}\left[\frac{5}{2}, \frac{\delta^2+2}{2}\right] + 6q_0^2\mathrm{B}\left[\frac{3}{2}, \frac{\delta^2+2}{2}\right] + q_0^4\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]
\end{aligned}$$

Using Equation (2.4.11) on the last line gives

$$\begin{aligned}
\boxed{\mathrm{A}} &= \mathrm{B}\left[\frac{5}{2}, \frac{\delta^2+2}{2}\right] + 6q_0^2\mathrm{B}\left[\frac{3}{2}, \frac{\delta^2+2}{2}\right] + q_0^4\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right] \\
&= \left(\frac{\frac{3}{2}}{\frac{3}{2}+\frac{\delta^2+2}{2}} + 6q_0^2\right)\mathrm{B}\left[\frac{3}{2}, \frac{\delta^2+2}{2}\right] + q_0^4\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right] \\
&= \left[\left(\frac{3}{\delta^2+5} + 6q_0^2\right)\left(\frac{1}{\delta^2+3}\right) + q_0^4\right]\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]
\end{aligned}$$

Using a similar argument to calculate $\boxed{\mathrm{B}}$ gives:

$$\boxed{\mathrm{B}} = \left(\frac{1}{\delta^2+3} + q_0^2\right)\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]$$

Therefore,

$$\begin{aligned}
\langle Q^2\rangle = \delta^2\frac{\boxed{\mathrm{A}}}{\boxed{\mathrm{B}}} &= \delta^2\frac{\left[\left(\frac{3}{\delta^2+5} + 6q_0^2\right)\left(\frac{1}{\delta^2+3}\right) + q_0^4\right]\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]}{\left(\frac{1}{\delta^2+3} + q_0^2\right)\mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]} \\
&= \delta^2\frac{\left[\left(\frac{3}{\delta^2+5} + 6q_0^2\right)\left(\frac{1}{\delta^2+3}\right) + q_0^4\right]}{\left(\frac{1}{\delta^2+3} + q_0^2\right)}
\end{aligned} \tag{2.4.13}$$

End-to-end chain length and radius of gyration

In [14, Chapter 11], it is shown that, for a bead-spring chain of N beads and $N-1$ springs, with spring connector vectors $\mathbf{Q}_i, i = 1, \ldots, (N-1)$, obeying the same force laws, the average equilibrium value for the squared length of the chain end-to-end vector (that is, the vector connecting the beads at each end of the chain) is:

$$\begin{aligned}
\langle R^2 \rangle &= \left\langle \sum_{i=1}^{N-1} \mathbf{Q}_i \right\rangle = \int \sum_i \sum_j \mathbf{Q}_i \cdot \mathbf{Q}_j \, \psi_{eq}(\mathbf{Q}^{N-1}) \, \mathrm{d}\mathbf{Q}^{N-1} \\
&= \sum_{i=1}^{N-1} \int Q_i^2 \prod_{j=1}^{N-1} \psi_{eq}(\mathbf{Q}^{N-1}) \, \mathrm{d}\mathbf{Q}^{N-1} \\
&= \sum_{i=1}^{N-1} \int Q_i^2 \prod_{j=1}^{N-1} \frac{\exp\left(-\phi\left(\mathbf{Q}_j\right)\right)}{\int \exp\left(-\phi\left(\mathbf{Q}_j\right)\right) \mathrm{d}\mathbf{Q}_j} \, \mathrm{d}\mathbf{Q}^{N-1} \\
&= (N-1) \times \int Q^2 \, \psi_{eq}\left(\mathbf{Q}\right) \, \mathrm{d}\mathbf{Q} = (N-1)\langle \mathbf{Q}^2 \rangle
\end{aligned}$$

This is derived from a combination of the results described in Bird et al. [14, Equations 11.3-22 and 11.4-2] and using Equation (2.4.4) for a chain of spring vectors.

Furthermore, let $\mathbf{r}_\nu, \nu = 1, \ldots, N$ denote the bead position vectors and define the chain centre-of-mass as $\mathbf{r}_c = \sum_{\nu=1}^{N} \mathbf{r}_\nu$. Then the radius of gyration of the chain becomes [14, Equation 11B.1-3]:

$$\begin{aligned}
\langle Rg^2 \rangle &= \left\langle \frac{1}{N} \sum_{\nu=1}^{N} (\mathbf{r}_\nu - \mathbf{r}_c) \cdot (\mathbf{r}_\nu - \mathbf{r}_c) \right\rangle \\
&= \frac{1}{N} \sum_{\nu=1}^{N} \int (\mathbf{r}_\nu - \mathbf{r}_c) \cdot (\mathbf{r}_\nu - \mathbf{r}_c) \, \psi_{eq} \, \mathrm{d}\mathbf{Q}^{N-1} \\
&= \frac{1}{6} \frac{N+1}{N} \langle R^2 \rangle = \frac{N^2 - 1}{6N} \langle \mathbf{Q}^2 \rangle
\end{aligned}$$

Note that spring connector vectors, $\mathbf{Q}_i, i = 1, \ldots, (N-1)$, can be calculated from bead position vectors using $\mathbf{Q}_i = \mathbf{r}_{i+1} - \mathbf{r}_i$.

Fourth moment

The equilibrium average of the fourth moment is given by:

$$\langle \mathbf{Q}^4 \rangle = \langle Q^4 \rangle = \frac{\int_{Q_0-\delta}^{Q_0+\delta} Q^6 e^{-\phi}\, \mathrm{d}Q}{\int_{Q_0-\delta}^{Q_0+\delta} Q^2 e^{-\phi}\, \mathrm{d}Q} \tag{2.4.14}$$

Using the same notation and similar calculation steps as for the second moment,

$$\begin{aligned}\langle Q^4 \rangle &= \frac{\int_{q_0-1}^{q_0+1} \delta^6 q^6 \left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}} \frac{\mathrm{d}Q}{\mathrm{d}q}\, \mathrm{d}q}{\int_{q_0-1}^{q_0+1} \delta^2 q^2 \left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}} \frac{\mathrm{d}Q}{\mathrm{d}q}\, \mathrm{d}q} \\ &= \delta^4 \frac{\int_{q_0-1}^{q_0+1} q^6 \left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\, \mathrm{d}q}{\int_{q_0-1}^{q_0+1} q^2 \left(1-(q-q_0)^2\right)^{\frac{\delta^2}{2}}\, \mathrm{d}q} = \delta^4 \frac{\boxed{\mathrm{C}}}{\boxed{\mathrm{B}}}\end{aligned}$$

Calculating $\boxed{\mathrm{C}}$ using the same approach as for $\boxed{\mathrm{A}}$ and $\boxed{\mathrm{B}}$ gives:

$$\boxed{\mathrm{C}} = \left[\left[\left(\frac{5}{\delta^2+7} + 15q_0^2\right)\left(\frac{3}{\delta^2+5}\right) + 15q_0^4\right]\left(\frac{1}{\delta^2+3}\right) + q_0^6\right] \mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]$$

Therefore,

$$\begin{aligned}\langle Q^4 \rangle = \delta^4 \frac{\boxed{\mathrm{C}}}{\boxed{\mathrm{B}}} &= \delta^4 \frac{\left[\left[\left(\frac{5}{\delta^2+7} + 15q_0^2\right)\left(\frac{3}{\delta^2+5}\right) + 15q_0^4\right]\left(\frac{1}{\delta^2+3}\right) + q_0^6\right] \mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]}{\left(\frac{1}{\delta^2+3} + q_0^2\right) \mathrm{B}\left[\frac{1}{2}, \frac{\delta^2+2}{2}\right]} \\ &= \frac{\delta^4 \left[\left[\left(\frac{5}{\delta^2+7} + 15q_0^2\right)\left(\frac{3}{\delta^2+5}\right) + 15q_0^4\right]\left(\frac{1}{\delta^2+3}\right) + q_0^6\right]}{\left(\frac{1}{\delta^2+3} + q_0^2\right)}\end{aligned} \tag{2.4.15}$$

Consistency checks with the FENE-Fraenkel second and fourth moments

In the limit of $\delta \to 0$, the extensibility of a FENE-Fraenkel spring diminishes. It should, therefore, behave like a rigid rod with length Q_0. Thus the expected values for the second and fourth moment in this situation should be $(Q_0)^2$ and $(Q_0)^4$.

For the second moment, taking $\delta \to 0$ in Equation (2.4.13),

$$\begin{aligned}
\langle Q^2 \rangle &= \delta^2 \frac{\left[\left(\frac{3}{\delta^2+5} + 6q_0^2\right)\left(\frac{1}{\delta^2+3}\right) + q_0^4\right]}{\left(\frac{1}{\delta^2+3} + q_0^2\right)} = \frac{\delta^2\left[\left(\frac{3}{\delta^2+5} + 6\frac{Q_0^2}{\delta^2}\right)\left(\frac{1}{\delta^2+3}\right) + \frac{Q_0^4}{\delta^4}\right]}{\left(\frac{1}{\delta^2+3} + \frac{Q_0^2}{\delta^2}\right)} \\
&= \frac{\delta^4\left[\left(\frac{3}{\delta^2+5} + 6\frac{Q_0^2}{\delta^2}\right)\left(\frac{1}{\delta^2+3}\right) + \frac{Q_0^4}{\delta^4}\right]}{\left(\frac{\delta^2}{\delta^2+3} + Q_0^2\right)} \\
&= \frac{\left[\left(3\frac{\delta^2}{\delta^2+5} + 6Q_0^2\right)\left(\frac{\delta^2}{\delta^2+3}\right) + Q_0^4\right]}{\left(\frac{\delta^2}{\delta^2+3} + Q_0^2\right)} \\
&= \frac{\left[\left(3\left(1 - \frac{5}{\delta^2+5}\right) + 6Q_0^2\right)\left(1 - \frac{3}{\delta^2+3}\right) + Q_0^4\right]}{\left(1 - \frac{3}{\delta^2+3} + Q_0^2\right)} \\
&\underset{s\to 0}{\to} \frac{\left[\left(3\left(1 - \frac{5}{5}\right) + 6Q_0^2\right)\left(1 - \frac{3}{3}\right) + Q_0^4\right]}{\left(1 - \frac{3}{3} + Q_0^2\right)} = \frac{Q_0^4}{Q_0^2} = Q_0^2
\end{aligned}$$

Similarly, for the fourth moment, taking $\delta \to 0$ in Equation (2.4.15) gives

$$\begin{aligned}
\langle Q^4 \rangle &= \frac{\delta^4 \left[\left[\left(\frac{5}{\delta^2+7} + 15q_0^2 \right) \left(\frac{3}{\delta^2+5} \right) + 15q_0^4 \right] \left(\frac{1}{\delta^2+3} \right) + q_0^6 \right]}{\left(\frac{1}{\delta^2+3} + q_0^2 \right)} \\
&= \frac{\delta^4 \left[\left[\left(\frac{5}{\delta^2+7} + 15\frac{Q_0^2}{\delta^2} \right) \left(\frac{3}{\delta^2+5} \right) + 15\frac{Q_0^4}{\delta^4} \right] \left(\frac{1}{\delta^2+3} \right) + \frac{Q_0^6}{\delta^6} \right]}{\left(\frac{1}{\delta^2+3} + \frac{Q_0^2}{\delta^2} \right)} \\
&= \frac{\delta^6 \left[\left[\left(\frac{5}{\delta^2+7} + 15\frac{Q_0^2}{\delta^2} \right) \left(\frac{3}{\delta^2+5} \right) + 15\frac{Q_0^4}{\delta^4} \right] \left(\frac{1}{\delta^2+3} \right) + \frac{Q_0^6}{\delta^6} \right]}{\left(\frac{\delta^2}{\delta^2+3} + Q_0^2 \right)} \\
&= \frac{\left[\left[\left(5\frac{\delta^2}{\delta^2+7} + 15Q_0^2 \right) \left(3\frac{\delta^2}{\delta^2+5} \right) + 15Q_0^4 \right] \left(\frac{\delta^2}{\delta^2+3} \right) + Q_0^6 \right]}{\left(\frac{\delta^2}{\delta^2+3} + Q_0^2 \right)} \\
&= \frac{\left[\left[\left(5\left(1 - \frac{7}{\delta^2+7} \right) + 15Q_0^2 \right) \left(3\left(1 - \frac{5}{\delta^2+5} \right) \right) + 15Q_0^4 \right] \left(1 - \frac{3}{\delta^2+3} \right) + Q_0^6 \right]}{\left(1 - \frac{3}{\delta^2+3} + Q_0^2 \right)} \\
&\underset{s \to 0}{\to} \frac{\left[\left[\left(5\left(1 - \frac{7}{7} \right) + 15Q_0^2 \right) \left(3\left(1 - \frac{5}{5} \right) \right) + 15Q_0^4 \right] \left(1 - \frac{3}{3} \right) + Q_0^6 \right]}{\left(1 - \frac{3}{3} + Q_0^2 \right)} = \frac{Q_0^6}{Q_0^2} \\
&= Q_0^4
\end{aligned}$$

These results agree with what is expected when using rods.

Furthermore, results for the second and fourth moments of FENE springs are also known [108]. These can be derived from the respective expressions for FENE-Fraenkel springs by setting $s = \sqrt{b}$ and $Q_0 = 0$ (so $q_0 = 0$). This gives

$$\langle Q^2 \rangle = \frac{b \left[\left(\frac{3}{b+5} + 0 \right) \left(\frac{1}{b+3} \right) + 0 \right]}{\left(\frac{1}{b+3} + 0 \right)} = \frac{3b}{b+5} \tag{2.4.16}$$

and

$$\langle Q^4 \rangle = \frac{b^2\left[\left[\left(\frac{5}{b+7}+0\right)\left(\frac{3}{b+5}\right)+0\right]\left(\frac{1}{b+3}\right)+0\right]}{\left(\frac{1}{b+3}+0\right)} = \frac{15b^2}{(b+5)(b+7)}. \tag{2.4.17}$$

These match known results calculated for the bead-spring chains consisting of FENE springs in Pham et al. [108].

For Sections 2.5 and 2.6 below, both dimensionalised and non-dimensionalised variables are mentioned. The non-dimensionalised parameters and variables are asterisked (*).

2.5 Excluded Volume Force Law

Excluded volume forces are applied between beads so as to prevent them occupying the same space [36]. Prakash and Öttinger [114] developed the following excluded volume potential, based on a Gaussian statistical distribution, for use in computer simulations:

$$E(\mathbf{Q}) = \frac{\nu k_B T}{(2\pi)^{\frac{3}{2}}\widetilde{d}^3}\exp\left(-\frac{\mathbf{Q}^2}{2\widetilde{d}^2}\right) \tag{2.5.1}$$

where,

- $\widetilde{d}$ is the standard deviation of the Gaussian distribution and represents the "width" of the excluded volume potential.
- ν represents the strength of the potential.

The repulsive force is then given by:

$$\mathbf{F} = -\frac{\partial E(\mathbf{Q})}{\partial \mathbf{Q}} \tag{2.5.2}$$

Following the approaches of [114] and [117], non-dimensionalisation of Equation (2.5.2) with respect to the same scales as for the spring force-laws gives:

$$\mathbf{F}^* = -\frac{\mathbf{Q}^* z^*}{(d^*)^5}\exp\left(-\frac{(\mathbf{Q}^*)^2}{2(d^*)^2}\right) \tag{2.5.3}$$

where

- $z^* = \nu(H/2\pi k_B T)^{\frac{3}{2}}$ is the dimensionless strength of the excluded volume (EV) repulsion.
- $d^* = \widetilde{d}\sqrt{H/k_B T}$ is the dimensionless range of the EV potential.

2.6 Hydrodynamic interactions

Hydrodynamic interactions (HI) are the intramolecular forces (in the case of a bead-spring chain, between beads of the same chain) that occur due to solvent interaction [66]. As with the excluded volume interactions, the strength of hydrodynamic interactions between beads depends on the distance between them. The non-dimensional hydrodynamic interaction parameter, h^*, is based on the bead radius and determines the threshold distance between beads for the strength of the interactions to become predominant.

To integrate these interaction into the model, the diffusion term in the equations governing the movement of beads through the fluid takes into account the distances between beads; the non-dimensionalised vector between the ν-th and μ-th beads is denoted $\mathbf{r}^*_{\nu\mu}$, with length $r^*_{\nu\mu}$. Furthermore, the strength of the interactions is related to the ratio between $r^*_{\nu\mu}$ and h^*.

The diffusion term therefore includes a tensor that is a function of the bead distances, $\zeta\boldsymbol{\Omega}$. The hydrodynamic interaction between the ν-th and μ-th beads is:

$$\zeta\boldsymbol{\Omega}_{\nu\mu} = \frac{3\sqrt{\pi}h^*}{4r^*_{\nu\mu}}\mathbf{C}\left(\mathbf{r}^*_{\nu\mu}\right) \tag{2.6.1}$$

where ζ is the drag coefficient of a bead in the chain through the fluid, and

$$\mathbf{C}\left(\mathbf{r}^*_{\nu\mu}\right) = \begin{cases} \left(1 + \dfrac{2\pi}{3}\dfrac{h^{*2}}{r^{*2}_{\nu\mu}}\right)\boldsymbol{\delta} + \left(1 - 2\pi\dfrac{h^{*2}}{r^{*2}_{\nu\mu}}\right)\mathbf{uu}, & \text{if } r^*_{\nu\mu} \geq 2\sqrt{\pi}h^* \\ \dfrac{r^*_{\nu\mu}}{2\sqrt{\pi}h^*}\left(\dfrac{8}{3} + \dfrac{3}{4}\dfrac{r^*_{\nu\mu}}{\sqrt{\pi}h^*}\right)\boldsymbol{\delta} + \dfrac{r^{*2}_{\nu\mu}}{8\pi h^{*2}}\mathbf{uu}, & \text{if } r^*_{\nu\mu} \leq 2\sqrt{\pi}h^* \end{cases} \tag{2.6.2}$$

where $\mathbf{u} = \mathbf{r}^*_{\nu\mu}/r^*_{\nu\mu}$.

This form of this tensor, known as the Rotne-Prager-Yamakawa tensor [123, 165] is explained in [112, §2].

2.7 Numerical method for solving the bead-spring configuration

The code contained in the appendices is based on a numerical method detailed in [112] and [48], based on a semi-implicit predictor-corrector scheme described in [102, §4.3.2] and developed further by Somasi et al. [139]. The principle is to solve for the bead configuration explicitly with respect to excluded volume interactions, but implicitly with respect to the spring forces. This has been shown by Öttinger [102] to lead to greater stability of the numerical algorithm, at the cost of significant increases in the computation time. A further step of converting the problem into a cubic polynomial that can be numerically solved helps to reduce the computation time. The steps are outlined below, but more details on the approach can be found in [112, §3].

2.7.1 Bead-Spring Configuration Equation

The aim is to obtain the chain configuration of a bead-spring chain consisting of N beads and $N - 1$ springs all following the same spring force law.

The mathematical basis of this problem is a stochastic differential equation, the *Fokker-Planck equation*, which describes the motion of a particle under the influence of random forces, such as in Brownian motion [102]. Let $\mathbf{X}_t$, be a random, n-dimensional vector, representing the positions of particles at time t, with

$$\mathrm{d}\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)\,\mathrm{d}t + \boldsymbol{\sigma}(\mathbf{X}_t, t)\,\mathrm{d}\mathbf{W}_t, \tag{2.7.1}$$

where

- $\boldsymbol{\mu}(\mathbf{X}_t, t) = (\mu_1, \ldots, \mu_n)$ is an n-dimensional vector representing the particle drift velocities.
- $\boldsymbol{\sigma}(\mathbf{X}_t, t)$ is an $n \times m$-dimensional matrix representing the diffusion of the particles, with associated diffusion tensor:

$$\mathbf{D}(\mathbf{x}, t) = \boldsymbol{\sigma}(\mathbf{x}, t)\boldsymbol{\sigma}^{\mathrm{T}}(\mathbf{x}, t). \tag{2.7.2}$$

- $\mathbf{W}_t$ is a *Wiener process*, a stochastic process representing Brownian motion. More details can be found in work by Öttinger [102] and Oksendal [100].

The probability density function, $p(\mathbf{x}, t)$, of $\mathbf{X}_t$, satisfies the Fokker-Planck equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i} \left[\mu_i(\mathbf{x}, t)\, p(\mathbf{x}, t)\right] + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2}{\partial x_i\, \partial x_j} \left[D_{ij}(\mathbf{x}, t)\, p(\mathbf{x}, t)\right]. \tag{2.7.3}$$

It can be shown that, for a bead-spring chain, the non-dimensionalised form of Equation (2.7.1) is equivalent to [56, 102, 117]:

$$\mathrm{d}\mathbf{R} = \left(\mathbf{K} \cdot \mathbf{R} + \frac{1}{4}\mathbf{D} \cdot \mathbf{F}^{\phi}\right) \mathrm{d}t + \frac{1}{\sqrt{2}}\mathbf{B} \cdot \mathrm{d}\mathbf{W}, \tag{2.7.4}$$

where

- $\mathbf{R}$ is the chain configuration vector consisting of the $3N$ co-ordinates of the dimensionless position vectors of the N beads, with $R_i = r_\nu^\alpha$, where $i = 3(\nu - 1) + \alpha$, $\nu = 1, \ldots, N$ and $\alpha = 1, 2, 3$.
- $\mathbf{F}^{\phi} = \mathbf{F}^{E} + \mathbf{F}^{S}$ is the force vector consisting of the $3N$ components of all the non-dimensionalised forces $F_\nu^\phi, \nu = 1, ..., N$, with
  - $\mathbf{F}^{E}$ corresponding to forces due to excluded volume interactions.
  - $\mathbf{F}^{S}$ corresponding to the forces of the springs acting on the beads — the net spring force on the ν-th bead is $\mathbf{F}_\nu^S = \mathbf{F}_\nu^c - \mathbf{F}_{\nu-1}^c$, where $\mathbf{F}_\nu^c$ is the tension in the spring connecting the $(\nu + 1)$th and the νth beads. In what follows:
    - $\mathbf{F}^{S}$ is expressed as a function of the whole chain configuration $\mathbf{R}$.
    - $\mathbf{F}_\nu^S$ is also a function of the whole chain configuration $\mathbf{R}$, but specifically deals with the spring forces on bead ν.
    - $\mathbf{F}_\nu^c$ is expressed as a function of the spring vector between these two beads, $\mathbf{Q}_\nu = \mathbf{r}_{\nu+1} - \mathbf{r}_\nu$. The expression for $\mathbf{F}^c$ is the spring force law, such as those given in Section 2.2.
- $\mathbf{K}$ is a block matrix consisting of $N \times N$ blocks of 3×3 matrices. The diagonal 3×3 blocks are each equal to the tensor $\boldsymbol{\kappa}$, the transposed non-dimensionalised velocity

gradient of the flow, while the rest of the off-diagonal blocks are equal to 0. The tensor $\boldsymbol{\kappa}$ takes different forms depending on the flow type being simulated. In this work,

- For a simple shear flow (as depicted in Figure 1.11),

$$\boldsymbol{\kappa} = \dot{\gamma} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.7.5}$$

  where $\dot{\gamma}$ is the constant, non-negative, non-dimensional shear rate.

- For a uniaxial extensional flow (where a "stretch" is applied along one axis and compression in the other two axes),

$$\boldsymbol{\kappa} = \dot{\varepsilon} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1/2 & 0 \\ 0 & 0 & -1/2 \end{pmatrix} \tag{2.7.6}$$

  where $\dot{\varepsilon}$ is the constant, non-negative, non-dimensional elongation rate.

- For a planar extensional flow (where the fluid is "stretched" along one axis and compressed on only one perpendicular axis; no strain is applied on the third axis),

$$\boldsymbol{\kappa} = \dot{\varepsilon} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.7.7}$$

  where $\dot{\varepsilon}$ is the constant, non-negative, non-dimensional elongation rate.

- **D** is a diffusion, block matrix consisting of $N \times N$ blocks containing the components of the $\boldsymbol{\Upsilon}_{\nu\mu}$ tensors, defined by:

$$\boldsymbol{\Upsilon}_{\nu\mu} = \delta_{\nu\mu}\boldsymbol{\delta} + \zeta\boldsymbol{\Omega}_{\nu\mu}, \tag{2.7.8}$$

where

  - $\delta_{\nu\mu}$ is the Kronecker delta which is equal to 1 when $\nu = \mu$ and 0 otherwise.
  - $\boldsymbol{\delta}$ is the unit tensor.
  - ζ is the drag coefficient of a bead in the chain through the fluid.
  - $\boldsymbol{\Omega}_{\nu\mu}$ is a tensor representing the hydrodynamic interactions between the ν-th and μ-th beads. It is a function of a hydrodynamic interaction parameter h^* and the vector connecting the ν-th and μ-th beads, $\mathbf{r}^*_{\nu\mu}$. This tensor is defined by Equations (2.6.1) and (2.6.2) in Section 2.6.

- $\mathbf{B}$ is a matrix such that $\mathbf{D} = \mathbf{B} \cdot \mathbf{B}^{\mathrm{T}}$.
- $\mathbf{W}$ is a $3N$-dimensional Wiener process.

The aim is to numerically solve Equation (2.7.4) to obtain the chain configuration vector R at different time points. To do this, it needs to be discretised; that is, instead of an equation considered over a time continuum, we solve for R at discrete time points.

2.7.2 Discretisation — Step 1 — First guess

The first step in discretising Equation (2.7.4) is to create a first guess of $\mathbf{R}$ at time-step $n+1$, $\widetilde{\mathbf{R}}_{n+1}$, given the value of $\mathbf{R}$ at time-step n, $\mathbf{R}_n$. This is defined by [112, Equation 14]

$$\widetilde{\mathbf{R}}_{n+1} = \mathbf{R}_n + \left(\mathbf{K} \cdot \mathbf{R}_n + \frac{1}{4}\mathbf{D}_n \cdot \mathbf{F}^S(\mathbf{R}_n) + \frac{1}{4}\mathbf{D}_n \cdot \mathbf{F}^E(\mathbf{R}_n)\right)\Delta t + \frac{1}{\sqrt{2}}\mathbf{B}_n \cdot \Delta\mathbf{W}_n \tag{2.7.9}$$

Components of the stochastic Wiener process vector term $\Delta\mathbf{W}_n$ are calculated using Equation (2.7.10) below, based on the probability integral transform.

$$X = \sqrt{\Delta t}\left(Y - \frac{1}{2}\right)\left[14.14855378\left(Y - \frac{1}{2}\right)^2 + 1.21569221\right], \tag{2.7.10}$$

where Y is a random number from a uniform distribution on $[0, 1]$ [102, Equation 3.125] [112, Equation 15].

2.7.3 Discretisation — Step 2 — Updated guess

The next step is to update the guess for $\mathbf{R}_{n+1}$.

$$\begin{aligned}\mathbf{R}_{n+1} = \mathbf{R}_n^* + \frac{\Delta t}{2}\left(\mathbf{K}\cdot\mathbf{R}_n + \mathbf{K}\cdot\widetilde{\mathbf{R}}_{n+1}\right) + \frac{\Delta t}{8}\mathbf{D}_n\cdot\left(\mathbf{F}^S(\mathbf{R}_n) + \mathbf{F}^S(\mathbf{R}_{n+1})\right)\\ + \frac{\Delta t}{8}\left(\mathbf{F}^E(\mathbf{R}_n) + \mathbf{F}^E(\widetilde{\mathbf{R}}_{n+1})\right) + \frac{1}{\sqrt{2}}\mathbf{B}_n\cdot\Delta\mathbf{W}_n\end{aligned} \tag{2.7.11}$$

Here, the excluded volume force term is treated explicitly as a function of our first guess, $\widetilde{\mathbf{R}}_{n+1}$, but the spring force term is treated *implicitly* as a function of the bead configuration that we are solving for, $\mathbf{R}_{n+1}$.

2.7.4 Discretisation — Step 3 — Recasting the updated guess

The next step introduces an operator $\mathcal{D}_\nu$. This takes the $(\nu+1)$-th block of 3 rows of a $3N\times M$ matrix (rows $3\nu+1$, $3\nu+2$ and $3\nu+3$) and subtracts the ν-th block of 3 rows (rows $3\nu-2$, $3\nu-1$ and 3ν) from it, resulting in a $3\times M$ matrix. As an example,

$$\mathcal{D}_\nu(\mathbf{R}) = \mathbf{r}_{\nu+1} - \mathbf{r}_\nu = \mathbf{Q}_\nu. \tag{2.7.12}$$

So for each bead ν, we can use this operation to get the spring connector vector $\mathbf{Q}_\nu$ from the bead configuration $\mathbf{R}$.

Applying this operator to Equation (2.7.11) gives:

$$\mathbf{Q}_{\nu,n+1} = \mathcal{D}_\nu\left[\widetilde{\Upsilon}_{n+1}\right] + \frac{1}{8}\mathcal{D}_\nu\left[\mathbf{D}_n\cdot\mathbf{F}^S(\mathbf{R}_{n+1})\right]\Delta t, \tag{2.7.13}$$

where,

$$\begin{aligned}\widetilde{\Upsilon}_{n+1} = \mathbf{R}_n + \left[\frac{1}{2}\left(\mathbf{K}\cdot\mathbf{R}_n + \mathbf{K}\cdot\widetilde{\mathbf{R}}_{n+1}\right) + \frac{1}{8}\mathbf{D}_n\cdot\mathbf{F}^S(\mathbf{R}_n)\right.\\ \left.+\frac{1}{8}\left(\mathbf{F}^E(\mathbf{R}_n) + \mathbf{F}^E(\widetilde{\mathbf{R}}_{n+1})\right)\right]\Delta t + \frac{1}{\sqrt{2}}\mathbf{B}_n\cdot\Delta\mathbf{W}_n\end{aligned} \tag{2.7.14}$$

The key point here is that nothing in $\widetilde{\Upsilon}_{n+1}$ is dependent upon $\mathcal{D}_\nu(\mathbf{R}_{n+1}) = \mathbf{Q}_{\nu,n+1}$. Furthermore, upon expansion of the second term on the RHS of Equation (2.7.13), we find a term

of the form

$$-\frac{1}{4}\left(\mathbf{F}^S_{\nu+1} - \mathbf{F}^S_{\nu}\right)\Delta t = -\frac{1}{4}\mathbf{F}^c_{\nu}\Delta t$$

Taking this term to the LHS of Equation (2.7.13) gives

$$\mathbf{Q}_{\nu,n+1} + \frac{1}{4}\mathbf{F}^c_{\nu}\Delta t = \mathcal{D}_{\nu}\left[\widetilde{\Upsilon}_{n+1}\right] + \frac{1}{8}\mathcal{D}_{\nu}\left[\mathbf{D}_n \cdot \mathbf{F}^S(\mathbf{R}_{n+1})\right]\Delta t + \frac{1}{4}\mathbf{F}^c_{\nu}\Delta t. \tag{2.7.15}$$

The final modification to make is to implement an iterative scheme whereby calculations on the lengths of each spring in the chain are calculated and are used to update the first guess of the bead configuration $\widetilde{\mathbf{R}}_{n+1}$ whilst still remaining at the same time point. So, for iteration index j, the equation to be solved now looks likes this:

$$\mathbf{Q}^{(j)}_{\nu,n+1} + \frac{1}{4}\mathbf{F}^{c,(j)}_{\nu}\Delta t = \mathbf{\Gamma}^{(j-1)}_{\nu,n+1} \tag{2.7.16}$$

where

$$\mathbf{\Gamma}^{(j-1)}_{\nu,n+1} = \mathcal{D}_{\nu}\left[\widetilde{\Upsilon}_{n+1}\right] + \frac{1}{8}\mathcal{D}_{\nu}\left[\mathbf{D}_n \cdot \mathbf{F}^{S,(j-1)}(\mathbf{R}_{n+1})\right]\Delta t + \frac{1}{4}\mathbf{F}^{c,(j-1)}_{\nu}\Delta t. \tag{2.7.17}$$

To explain the iterative process, we quote from [112, §3]. The j-th iteration in this scheme consists of successively calculating all the $\mathbf{Q}^{(j)}_{\nu,n+1}$, starting from $\nu = 1$. For the first bead, $\mathbf{Q}^{(j)}_{1,n+1}$ is obtained by calculating $\mathbf{\Gamma}^{(j-1)}_{1,n+1}$ using the bead configuration $\mathbf{R}^{(j-1)}_{n+1}$ obtained from the previous iteration, and then solving Equation (2.7.16) exactly.

In the same j-th iteration, for the next bead, $\nu = 2$, we find $\mathbf{Q}^{(j)}_{2,n+1}$ by updating the bead configuration $\mathbf{R}^{(j-1)}_{n+1}$ with our newly found value of $\mathbf{Q}^{(j)}_{1,n+1}$ and leaving the rest of the spring connector vectors unchanged. This updated bead configuration is used to calculate $\mathbf{\Gamma}^{(j-1)}_{2,n+1}$ and then solve Equation (2.7.16) for $\mathbf{Q}^{(j)}_{2,n+1}$.

This process continues along the chain until the last connector vector, $\mathbf{Q}^{(j)}_{N-1,n+1}$ is calculated, whereupon we have our completed j-th iteration of the bead configuration at time $n+1$, $\mathbf{R}^{(j)}_{n+1}$, and move on to the next iteration, $j+1$, where we start the process again from the first bead. The iteration continues until the relative error $\left|\mathbf{R}^{(j)}_{n+1} - \mathbf{R}^{(j-1)}_{n+1}\right| / \left|\mathbf{R}^{(j)}_{n+1}\right|$ is below a certain tolerance. For reasons of algorithm stability [112], the algorithm starts with $\mathbf{R}^{(0)}_{n+1} = \mathbf{R}_n$, not $\widetilde{\mathbf{R}}_{n+1}$.

2.7.5 Discretisation — Step 4 — Generating a cubic

Observe that the LHS of Equation (2.7.16) is dependent only on $\mathbf{Q}_{\nu,n+1}^{(j)}$ being sought and the RHS, $\mathbf{\Gamma}_{\nu,n+1}^{(j-1)}$, is based on $\mathbf{Q}_{\nu,n+1}^{(j-1)}$ and $\mathbf{R}_{n+1}^{(j-1)}$ which are known from the previous iteration. Thus we are left to find $\mathbf{Q}_{\nu,n+1}^{(j)}$ by solving Equation (2.7.16). For the rest of this section, we shall simplify the notation so that this equation now reads:

$$\mathbf{Q} + \frac{\Delta t}{4}\mathbf{F} = \mathbf{\Gamma}. \tag{2.7.18}$$

Recall that $\mathbf{F}$ takes the form of a non-dimensionalised spring force function of a spring connector vector. If we replace $\mathbf{F}$ with the expression for the FENE force law as in Equation (2.2.1), we get

$$\mathbf{Q} + \frac{\Delta t}{4}\frac{\mathbf{Q}}{1-\frac{\mathbf{Q}^2}{b}} = \mathbf{Q}\left(1 + \frac{\Delta t}{4}\frac{1}{1-\frac{\mathbf{Q}^2}{b}}\right) = \mathbf{\Gamma}. \tag{2.7.19}$$

Observe that $\mathbf{\Gamma}$ is a scalar multiple of $\mathbf{Q}$, so this equation still holds if we replace these vectors with $\Gamma = |\mathbf{\Gamma}|$ and $Q = |\mathbf{Q}|$. If we choose $x = \frac{|\mathbf{Q}|}{\sqrt{b}}$, we get:

$$\sqrt{b}x\left(1 + \frac{\Delta t}{4}\frac{1}{1-x^2}\right) = \Gamma. \tag{2.7.20}$$

This can be rearranged into the following cubic polynomial equation:

$$x^3 - \tilde{\Gamma}x^2 - \left(1 + \frac{\Delta t}{4}\right)x + \tilde{\Gamma} = 0, \tag{2.7.21}$$

where we have used $\tilde{\Gamma} = \frac{\Gamma}{\sqrt{b}}$.

In the case of the FENE-Fraenkel force law, following the same method but replacing $\mathbf{F}$ with the expression in Equation (2.2.3) gives us the following polynomial equation:

$$x^3 + \left(-2y - \tilde{\Gamma}\right)x^2 + \left(y^2 + 2\tilde{\Gamma}y - \frac{\Delta t}{4} - 1\right)x + \left(\frac{\Delta t}{4}y + \tilde{\Gamma} - \tilde{\Gamma}y^2\right) = 0 \tag{2.7.22}$$

where $x = \frac{Q}{\delta}$, $\tilde{\Gamma} = \frac{\Gamma}{\delta}$ and $y = \frac{Q_0}{\delta}$. As a consistency check, setting $Q_0 = 0$ (*i.e.* $y = 0$) regenerates the equation for the FENE case.

To check that this polynomial does have a root in the interval $(y-1, y+1)$ (which corre-

sponds to the scaled length range of a FENE-Fraenkel spring), we can use the Intermediate Value Theorem. If we let the polynomial be represented by a function $f : \mathbb{R} \to \mathbb{R}$, then we wish to solve $f(x) = 0$ for x. Since it is a polynomial, f is continuous. We can also show that $f(y-1) = \frac{\Delta t}{4}$ and $f(y+1) = -\frac{\Delta t}{4}$. By the Intermediate value theorem, this polynomial has a root between $y-1$ and $y+1$.

2.8 Summary

In this chapter, the FENE-Fraenkel spring for bead-spring chain simulations has been discussed. Analytical expressions of the second and fourth moments for a FENE-Fraenkel spring have been derived, and have been shown to be consistent with known results for the FENE spring and in the limit of the extensibility δ tending to zero, where we expect the spring to behave as a rigid rod. The second moment (Equation (2.4.13)) is particular important in establishing the length of a chain (Subsection 2.4.3) and can therefore be used to compare against data obtained from simulations for the length of bead-spring chains made from FENE-Fraenkel springs.

An algorithm, devised by Ranganathan and Prakash [117], for simulating polymers as bead-spring chains in fluid flows, using different types of fluid flow and incorporating excluded volume and hydrodynamic interactions has also been outlined. This forms the basis of the simulation programme used to produce results in Chapter 3.

Chapter 3

Simulating polymers in flow with FENE-Fraenkel springs

In this chapter, we consider the simulation of bead-spring chains in fluid flows. The goal of this simulation work was to see if molecules in extensional and shear flows could be simulated using bead-spring chains with FENE-Fraenkel springs along with hydrodynamic interactions, as described in the previous chapter. As an initial test of the code, some simulation outputs were verified against analytical properties of FENE-Fr. springs derived in the previous chapter. The simulator code's performance was then compared against results published by Hsieh et al. [48], who had also investigate FENE-Fraenkel springs but using a slightly different parameter space. Finally, tests with bead-spring chains and FENE-Fraenkel springs in fluid flow and with or without hydrodynamic interactions were performed and the orientation parameter of the chains was studied.

3.1 Code alterations to incorporate the FENE-Fraenkel spring

A program constructed from Fortran code, provided by Prof. Ravi Jagadeeshan, was modified to implement the model described in the previous chapter. It was originally built to run simulations of single bead-spring chains using a variety of spring force laws and in different types of fluid flow. Details of the Fortran code as originally provided are in

Appendix B, along with the code itself in Appendix D. The changes made to the code to incorporate the FENE-Fraenkel spring are documented in full in Appendix C, but a summary of the main changes is provided below.

- A new parameter to incorporate the non-zero natural length Q_0 of a FENE-Fraenkel spring (compared to the "zero natural length" of a FENE spring) was added to the code.
- The FENE-Fraenkel spring force law and its associated properties, such as the spring potential ϕ, were added to the code.
- Existing variables which were previously storing values with "single-precision" (32-bit storage) were modified to store values with "double-precision" (64-bit storage). This is due to the added sensitivity of the spring force to small changes in spring length, as any small deviations are now compared against the sum of the natural length and maximum extensibility, which is a much larger value than the maximum extensibility alone, as in the FENE spring case. This sensitivity can be seen by how close the roots of the cubic polynomial (Equation (2.7.22)) are to each other. Other variables were changed to double-precision, to allow functions and subroutines to communicate with each other without a loss in precision.
- Solutions to the cubic polynomial Equation (2.7.22) were previously found by making an initial guess and using the Newton-Raphson numerical method to converge towards a root. For the FENE-Fraenkel spring, there is a risk that more than one of the three roots for the cubic polynomial could be treated as a valid length for a spring. Furthermore, the choice of the initial guess for the Newton-Raphson method affects if there is convergence to a root and which root is the limit of the procedure. To avoid these problems, this was replaced with a cubic solver function based on an algorithm in Press et al. [115, §5.6] to directly calculate the roots of the cubic polynomial. The algorithm would first check that any root was within the permitted range of spring lengths, namely $(Q_0 - \delta, Q_0 + \delta)$. If more than one root was proven to be in this range, then the algorithm would select the root closest to the previous spring length.

3.2 Verification of simulation against analytically derived properties

In Section 2.4, formulae for a number of properties of a FENE-Fraenkel spring in zero shear rate equilibrium conditions were derived. To ascertain if the code outputs correspond with values predicted in the theory, simulations were run with the parameters given in Table 3.1.

Table 3.1: Parameters used for simulation testing to see if analytical results for the FENE-Fraenkel are obtained.

Variable	Inputs
Number of beads, N	2
Shear strain rates, $\dot{g}$	$0, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}$
Time steps, Δt	1, 2, 4
Pairs of spring natural length and extensibility parameters, (Q_0, δ)	(100, 1), (100, 5), (200, 1), (200, 5), (200, 10)
Run time	10000
Number of samples	10000

3.2.1 Zero-shear conditions

For zero shear strain rate cases, data were obtained for the squared end-to-end chain length, $\langle R^2 \rangle$. Since only 2 beads and one spring were being simulated, this value is equal to the single spring second moment, $\langle Q^2 \rangle$.

Table 3.2 compares the analytically-derived values for $\langle R^2 \rangle$ against the average values of the mean and standard error from simulation data where different time-step sizes were used. The simulation means and standard errors in this table were calculated by taking the average of the mean value and standard error obtained for the spring property at each time point, across all time points. That is, if property p is the variable of interest, and, for a particular simulation i at time point t, this has value $p_{i,t}$, then

$$\text{Standard error} = \sqrt{\frac{\left(\sum_{i,t} p_{i,t} - \bar{p}\right)^2}{n(n-1)}}, \tag{3.2.1}$$

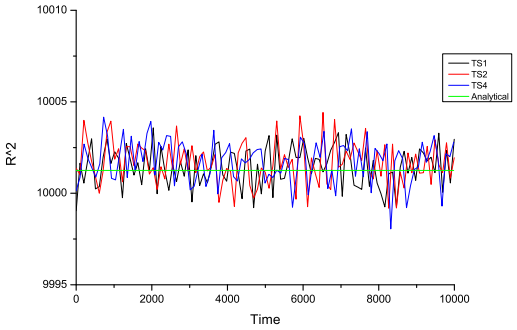
where the mean $\bar{p}$ is:

$$\text{Mean, } \bar{p} = \frac{\sum_{i,t} p_{i,t}}{n}.$$

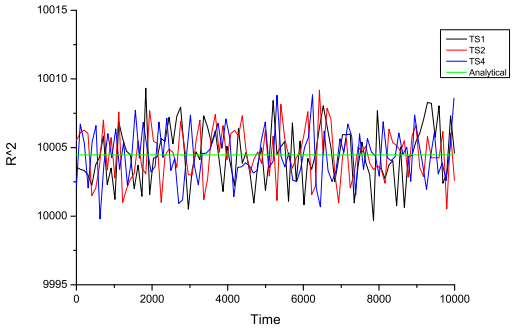
and n is the product of the number of simulated springs and number of timepoints at which the spring property is measured (i.e. the total number of individual measurements made of property p).

The tabulated data suggests that, under zero shear conditions, the simulations behave close to what is predicted by the analytical results derived in the previous chapter. Differences between the mean values and the analytically derived values were observed to be larger with $Q_0 = 200$ than with $Q_0 = 100$ and with larger values of δ. However, these differences are well within the range of the standard error and are accountable to fact that increasing the size of Q_0 and δ make the spring longer and more flexible, which gives rise to the larger error.

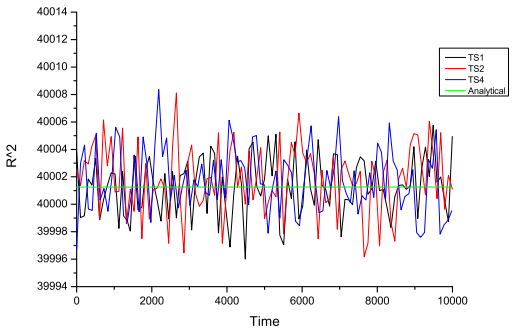
The time-step chosen does not appear to change the size of the standard error or the behaviour of the simulation with respect to the other chosen parameters. This can also be seen in Figure 3.1, which illustrates the measured values of $\langle R^2 \rangle$ in each of the different pairs of (Q_0, δ) values over time and for different time-step values. The oscillations of property values above and below the green line, marking the analytical value, do not change markedly with the time-step size chosen.



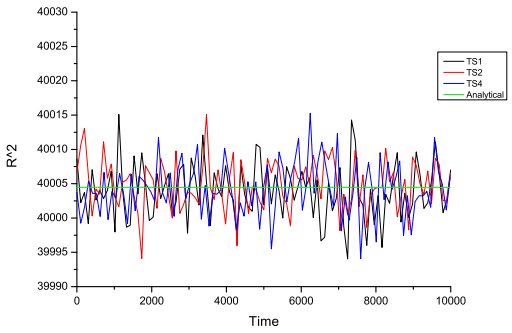
(a) $(Q_0, \delta) = (100, 1)$



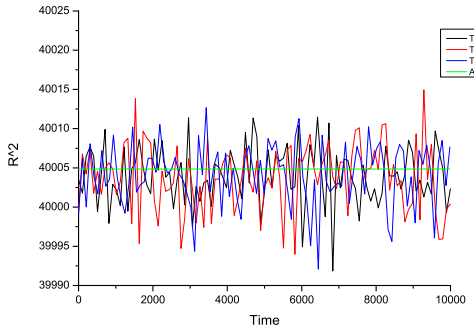
(b) $(Q_0, \delta) = (100, 5)$



(c) $(Q_0, \delta) = (200, 1)$



(d) $(Q_0, \delta) = (200, 5)$



(e) $(Q_0, \delta) = (200, 10)$

Figure 3.1: Plots of the squared spring length $\langle R^2 \rangle$ against time for different time-step values and different combinations of natural length Q_0 and extensibility δ under zero shear conditions and a line indicating the analytically derived value. Standard error bars have been ommitted for clarity.

Table 3.2: Mean values and standard errors for $\langle R^2 \rangle$ with zero shear rate. These have been averaged over all timepoints.

(Q_0, δ)	Analytical value	Simulation results		Time step
		Mean	Standard Error	
(100, 1)	10001.2	10001.5	2.3	1
		10001.7	2.4	2
		10001.7	2.4	4
(100, 5)	10004.5	10004.4	3.8	1
		10004.6	3.8	2
		10004.5	3.8	4
(200, 1)	40001.3	40001.3	4.6	1
		40001.8	4.7	2
		40001.7	4.8	4
(200, 5)	40004.5	40003.9	7.6	1
		40004.8	7.6	2
		40004.2	7.6	4
(200, 10)	40004.9	40004.0	7.9	1
		40003.8	7.9	2
		40004.1	7.9	4

3.2.2 Non-zero shear conditions

For non-zero shear strain rate cases, simulation data were obtained for the squared end-to-end chain length, $\langle R^2 \rangle$. Table 3.3 compares the analytically-derived values for $\langle R^2 \rangle$ under zero shear conditions against the average values of the mean and standard error from simulation data where different shear rates were used. For brevity, only data for simulations with a time-step size of 1 are displayed here. The simulation means and standard errors are calculated in a similar way to the data in Table 3.2, but only the time points of the last fifth of the simulation were used. This is because, as seen in Figure 3.2, any noticable change in spring property values are settled in this part of the simulation.

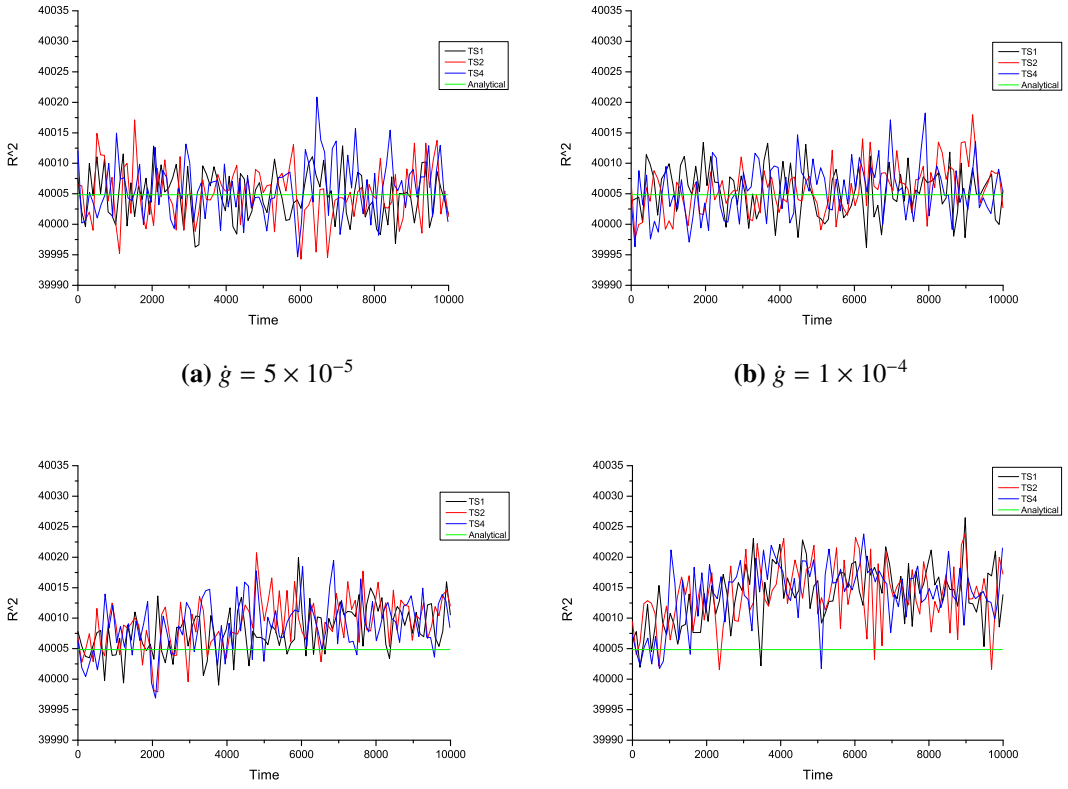
The data indicate that, for the choices of shear rate values below 4×10^{-4} and for most choices of parameter values, changes in the $\langle R^2 \rangle$ values observed in simulation due to the increased shear forces and the zero shear analytical values do not exceed the size of the standard error. However, when the shear rate of $\dot{g} = 4$ is used, simulated values of $\langle R^2 \rangle$ are markedly higher than the zero shear analytical values. This is particularly noticeable in cases where $Q_0 = 200$. This is also in keeping with the intuition that a longer and more flexible spring will extend further in high shear conditions.

Figure 3.2 illustrates the measured values of $\langle R^2 \rangle$ over time for $Q_0 = 200$ and $\delta = 10$,

under different shear rate and time-step values. As in the zero shear case, the time-step chosen does not appear to change the size of the standard error or the behaviour of the simulation with respect to the other chosen parameters; no plot for any particular time step appears to be smoother than the rest. Furthermore, increases in the shear rate raise the value of $\langle R^2 \rangle$ attained at 3000 time units, as seen in Figure 3.2 with an increase in the size of the gap between the simulation runs and the zero-shear analytical value (green line on each plot) with successive shear rate increases. Similar behaviour was observed with other combinations of (Q_0, δ) tested, but the effects of increasing the shear rate were observed to be weaker with smaller extensibility δ values.

Table 3.3: Mean values and standard errors for $\langle R^2 \rangle$ under different linear shear rates and using a time step of 1 time unit. These have been averaged over the timepoints of the last 2000 time units of the 10000 time unit length simulation.

(Q_0, δ)	Zero-shear analytical value	Simulation results		Shear rate $\dot{g}$ / 10^{-4}
		Mean	Standard Error	
(100, 1)	10001.2	10001.8	2.3	0.5
		10001.1	2.3	1
		10002.1	2.3	2
		10002.4	2.3	4
(100, 5)	10004.5	10004.3	3.8	0.5
		10004.2	3.8	1
		10005.0	3.8	2
		10006.2	3.8	4
(200, 1)	40001.3	40001.8	4.6	0.5
		40002.2	4.6	1
		40003.1	4.6	2
		40004.0	4.6	4
(200, 5)	40004.5	40005.7	7.6	0.5
		40005.2	7.6	1
		40009.7	7.5	2
		40012.3	7.6	4
(200, 10)	40004.9	40003.6	7.9	0.5
		40005.1	7.9	1
		40009.5	7.9	2
		40015.0	7.9	4



(a) $\dot{g} = 5 \times 10^{-5}$ **(b)** $\dot{g} = 1 \times 10^{-4}$

(c) $\dot{g} = 2 \times 10^{-4}$ **(d)** $\dot{g} = 4 \times 10^{-4}$

Figure 3.2: Plots of the squared spring length $\langle R^2 \rangle$ against time for different time-step values and different shear rates and a line indicating the analytically derived value under zero shear conditions, using a FENE-Fraenkel spring with natural length $Q_0 = 200$ and extensibility $\delta = 10$. Standard error bars have been ommitted for clarity. Means and standard errors for the last 2000 time units of each simulation are given in Table 3.3.

3.3 Comparison against work by Hsieh et al. [48]

The FENE-Fraenkel spring force law was first proposed by Hsieh et al. [48], with a discretisation protocol similar to that used for simulations described in this thesis (see Chapter 2). The code as documented here was tested against the data published in that paper as a further verification step.

3.3.1 Conversion between parameter values

In order to verify the outputs of the code used in this investigation against results reported by Hsieh et al. [48], adjustments were reuired to account for the differences in length, force and time scales used. In particular, the non-dimensionalisation of the FENE-Fraenkel force law is different between this work and the work of Hsieh et al. [48].

Recall that the dimensionalised form of the FENE-Fraenkel force law is:

$$\mathbf{F} = \frac{H\left(Q - Q_0\right)}{1 - \left(\frac{Q - Q_0}{\delta}\right)^2} \frac{\mathbf{Q}}{Q}$$

In this work, the non-dimensionalisation of this equation uses the following scales:

- $\sqrt{k_B T / H}$ — length scale, where k_B is Boltzmann's constant, T is absolute temperature, and H is the spring constant.
- $\zeta / 4H$ — time scale, where ζ is the drag coefficient of a bead in the bead spring chain.
- $\sqrt{k_B T H}$ — force scale.

This results in the following non-dimensionalised equation:

$$\mathbf{F}^* = \frac{\left(Q^* - Q_0^*\right)}{1 - \left(\frac{Q^* - Q_0^*}{\delta^*}\right)^2} \frac{\mathbf{Q}^*}{Q^*}, \tag{3.3.1}$$

In the work of [48], the following scales are used for non-dimensionalisation.

- Q_0 — length scale, where Q_0 is the natural length of a spring.

- $Q_0^2\zeta/k_BT$ — time scale.
- k_BT/Q_0 — force scale.

This results in the following non-dimensionalised equation:

$$\mathbf{F}^* = \frac{H^*(Q^*-1)}{1-\left(\frac{Q^*-1}{\delta^*}\right)^2}\frac{\mathbf{Q}^*}{Q^*}, \tag{3.3.2}$$

With the different non-dimensionalistion paradigms, there needs to be a means of converting values of H^* and δ^* used in [48] into values of Q_0^* and δ^* for the simulations performed for this work.

Observe that the spring constant, H, has units of force/distance. Therefore,

$$H^* = H \times \left(\frac{\text{Force scale}}{\text{Length scale}}\right)^{-1} = H \times \left(\frac{\frac{k_BT}{Q_0}}{Q_0}\right)^{-1} = \frac{HQ_0^2}{k_BT}.$$

Thus,

$$H = \frac{k_BTH^*}{Q_0^2} \tag{3.3.3}$$

Applying the non-dimensionalisation scheme of this work gives:

$$\begin{aligned} Q_0^* &= \frac{Q_0}{\text{Length scale}} = \frac{Q_0}{\sqrt{\frac{k_BT}{H}}} = Q_0 \times \left(\sqrt{\frac{k_BT}{\frac{k_BTH^*}{Q_0^2}}}\right)^{-1} \\ &= Q_0 \times \left(\sqrt{\frac{k_BTQ_0^2}{k_BTH^*}}\right)^{-1} = Q_0 \times \left(\frac{Q_0}{\sqrt{H^*}}\right)^{-1} = \sqrt{H^*} \end{aligned}$$

Furthermore, as $\delta = \delta^* \times \text{Length scale} = \delta^* \times Q_0$, this gives

$$\delta^* = \frac{\delta}{\text{Length scale}} = s \times \left(\sqrt{\frac{k_BT}{H}}\right)^{-1} = \delta^* \times Q_0 \times \left(\sqrt{\frac{k_BT}{\frac{k_BTH^*}{Q_0^2}}}\right)^{-1}$$
$$= \delta^* \times Q_0 \times \left(\sqrt{\frac{k_BTQ_0^2}{k_BTH^*}}\right)^{-1} = \delta^* \times Q_0 \times \left(\frac{Q_0}{\sqrt{H^*}}\right)^{-1} = \delta^* \times \sqrt{H^*}.$$

Conversion between the time interval, Δt^*, used by Hsieh et al. [48] and the time interval, Δt^*, used in the simulations documented in this thesis can also be deduced in a similar way.

$$\Delta t^* = \frac{\Delta t}{\frac{\zeta}{4H}} = \Delta t^* \times \frac{Q_0^2\zeta}{k_BT} \times \frac{4H}{\zeta} = \Delta t^* \times \frac{Q_0^2\zeta}{k_BT} \times \frac{4\frac{k_BTH^*}{Q_0^2}}{\zeta}$$
$$= \Delta t^* \times \frac{Q_0^2\zeta}{k_BT} \times \frac{4k_BTH^*}{Q_0^2\zeta} = \Delta t^* \times 4H^*.$$

Finally, a similar argument can be used to convert strain rates between the two paradigms. Let $\dot{g}^*$ and $\dot{g}^*$ be the non-dimensional strain rates in the work of [48] and in the simulations documented in this thesis respectively, with $\dot{g}$ as the corresponding dimensional strain rate. Noting that strain rates have units of reciprocal time,

$$\dot{g}^* = \dot{g} \times \frac{\zeta}{4H} = \frac{\dot{g}^*}{\frac{Q_0^2\zeta}{k_BT}} \times \frac{\zeta}{4H} = \dot{g}^* \times \frac{k_BT}{4Q_0^2H}$$
$$= \dot{g}^* \times \frac{k_BT}{4Q_0^2} \times \frac{Q_0^2}{k_BTH^*} = \frac{\dot{g}^*}{4H^*}.$$

Therefore, the following four equations can be used to convert parameters between the two non-dimensionalisation paradigms.

$$Q_0^* = \sqrt{H^*} \tag{3.3.4a}$$
$$\delta^* = \delta^* \times \sqrt{H^*}, \tag{3.3.4b}$$
$$\Delta t^* = \Delta t^* \times 4H^*. \tag{3.3.4c}$$
$$\dot{g}^* = \frac{\dot{g}^*}{4H^*}. \tag{3.3.4d}$$

Furthermore, Equations (3.3.4b) and (3.3.4c) can also be modified for converting other non-dimenionsalised lengths and times.

3.3.2 Choice of H^* and its influence on simulated polymer stresses

To investigate the effects of the spring constant parameter, H^*, on simulated values of first normal polymer stress coefficient, Hsieh et al. [48, §III. A.] used the following parameters:

- Number of beads = 11 — This is not explicitly stated, but implied by first paragraph of section II, part C in the paper.
- No hydrodynamic or excluded volume interactions.
- Values of H^* used are 10^2, 10^4 and 10^6.
- $\delta^* = 0.01$ and $\Delta t^* = 0.0001$.
- Results are averaged over 10000 individual, independent trajectories.
- Relaxation times τ_1 are calculated according to the following formula, attributed to Doyle et al. [29]:

$$\tau_1 = 0.00142(N_s + 1)^2, \tag{3.3.5}$$

  where N_s is the number of springs in the bead-spring chain (or rods in the bead-rod chain). Thus, in this instance,

$$\tau_1 = 0.00142 \times 11^2 = 1.7182 \text{ time units}$$

- The Weissenberg number is a non-dimensional number given by

$$\text{Wi} = \text{Strain Rate} \times \text{Relaxation Time}, \tag{3.3.6}$$

  where the relaxation time is calculated in Equation (3.3.5) and the strain is for a uniaxial extensional flow (see Subsection 2.7.1). In these simulations Wi = 10.

Therefore,

$$\text{Strain Rate } \dot{g}^* = \frac{10}{1.7182} \approx 5.82$$

- The full run time of the simulation is 4 time units.

Using Equation (3.3.4), the following parameters were used for simulations.

- Number of beads = 11.
- No hydrodynamic or excluded volume interactions.
- Results are averaged over 10000 individual, independent trajectories.
- The fluid flow is extensional and uniaxial.
- Values of the simulation run time, the time-step Δt^*, the natural length Q_0^*, the extensibility parameter δ^* and the extensional strain rate $\dot{g}^*$ are given in Table 3.4. These correspond to the different choices of H^*.

Table 3.4: Values of various parameters under the non-dimensionalisation paradigm of this work converted from the parameters used in [48] to analyse the effect of the choice of spring constant parameter on simulation results.

Simulation	Δt^*	Run Time	Q_0^*	δ^*	$\dot{g}^*$	H^*
A	0.04	8×10^2	10	0.1	1.455×10^{-2}	10^2
B	4	8×10^4	100	1	1.455×10^{-4}	10^4
C	400	8×10^6	1000	10	1.455×10^{-6}	10^6

The *polymer stress* of polymers in solution is given by

$$\boldsymbol{\sigma} = \rho \sum_{i=1}^{N_s} \langle \mathbf{F}_i^{sp} \mathbf{Q}_i \rangle, \tag{3.3.7}$$

where:

- ρ = the number density of polymer chains. This is set to one in both the results reported by Hsieh et al. [48] and in this work.
- N_s = the number of springs in the chain. In this case, $N_s = 10$.
- $\mathbf{F}_i^{sp}$ = the spring force vector in spring i.
- $\mathbf{Q}_i$ = the spring vector of spring i.

The first component of this expression, *i.e.*

$$\sigma_{11} = \sum_{i=1}^{N_s} F_{1,i}^{sp} Q_{1,i}, \tag{3.3.8}$$

is known as the *first normal polymer stress coefficient*. This value was calculated in these simulations and compared against the published results, where $F_{1,i}^{sp}$ and $Q_{1,i}$ are the first components of $\mathbf{F}_i^{sp}$ and $\mathbf{Q}_i$ respectively, with the uniaxial extensional flow stretching along this axis.

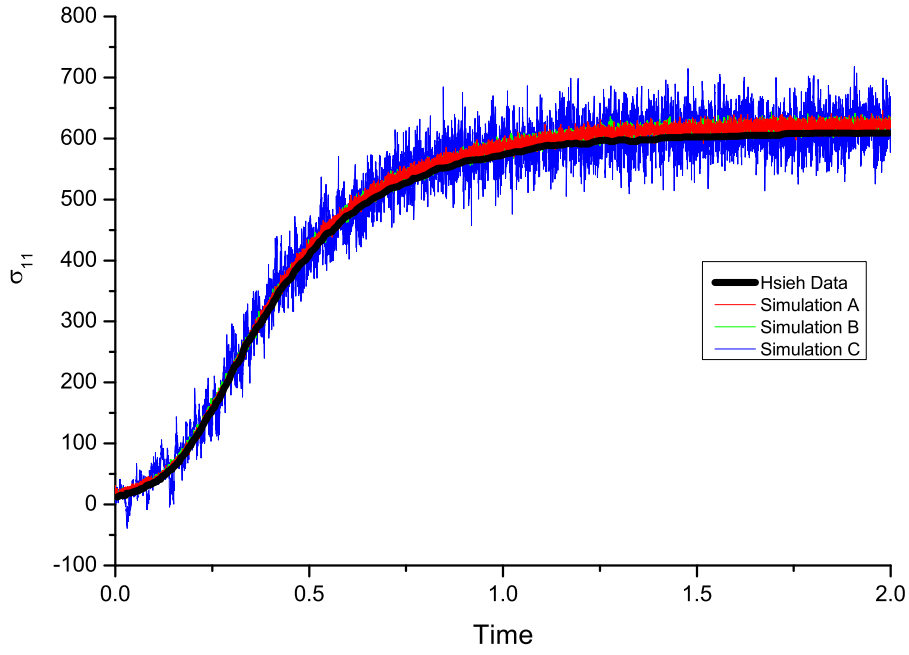


Figure 3.3: Plots of σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, corresponding to different parameter combinations described in Table 3.4. Time scales of the simulations have been rescaled to match that of the data given in Hsieh et al. [48], each corresponding to a different value of spring constant (spring stiffness). The "Hsieh data" plot corresponds to data obtained from Figure 3 in Hsieh et al. [48]

Figure 3.3 displays the plots of the first normal polymer stress coefficient, σ_{11}, against time for the three simulation parameter combinations and a black line representing the data from Hsieh et al. [48] (extracted using Plot Digitizer). These plots suggest that the use of $H^* = 10^6$ for the spring constant value (corresponding to the parameters for simulation C in Table 3.4) results in large fluctuations in the σ_{11} value over time, compared with the results obtained using $H^* = 10^2$ or 10^4 (simulations A and B). This was also observed by Hsieh et al. [48] in their data, and corresponds to the increased stiffness in the spring due

to the higher spring constant value. Furthermore, upon analysis of the spread of values for σ_{11} in the plateau (defined here to be between time $t = 1.8$ and $t = 2.0$ inclusive), a similar scale of deviation from the average value along the plateau is observed between our data and the data in Hsieh et al. [48] for $H^* = 10^2$ and 10^4, but the data for $H^* = 10^6$ is noisier than their results (see Figure 3.4 and [48, Figure 3 insert]).

Table 3.5 indicates that there is a small but noticeable discrepancy of approximately 2% between the values obtained by our simulations and the data published by Hsieh et al. [48]. This can also be seen by the black line in Figure 3.3 not lying in the middle of the range of values of σ_{11} covered by our simulations. In simulation C, the mean value from the data in [48] is within the standard error bar for the data obtained in our work. However, the error much larger than in simulations A and B, where the standard error is five times smaller, but the Hsieh data mean lies outside the bottom end of the error bar range. Further tests would need to performed based upon a a larger range of spring constant values to ascertain if the discrepancies observed are systematic errors. One possibility for why the difference could be in the way that the simulation programs for each source are coded. While both sources use an approach based upon the semi-implicit algorithm as devised by Öttinger [102], the differences in the non-dimensionalisation and reparametrisation of the bead-spring chain setup could be reflected in the different ways in which the simulations performed by Hsieh et al. [48] and for our research were programmed. The discrepancies in the plateau values shown in Table 3.5 could be the result of these differences.

Table 3.5: Averaged values for the mean and standard error of σ_{11} between time-points 1.8 and 2.0 (the "plateau") in simulations using parameters described in Table 3.4 and in [48, Figure 3].

Hsieh data mean	Simulation results		Simulation	H^*
	Mean	Standard Error		
608.7	621.7	13.5	A	10^2
	621.7	12.8	B	10^4
	616.6	61.3	C	10^6

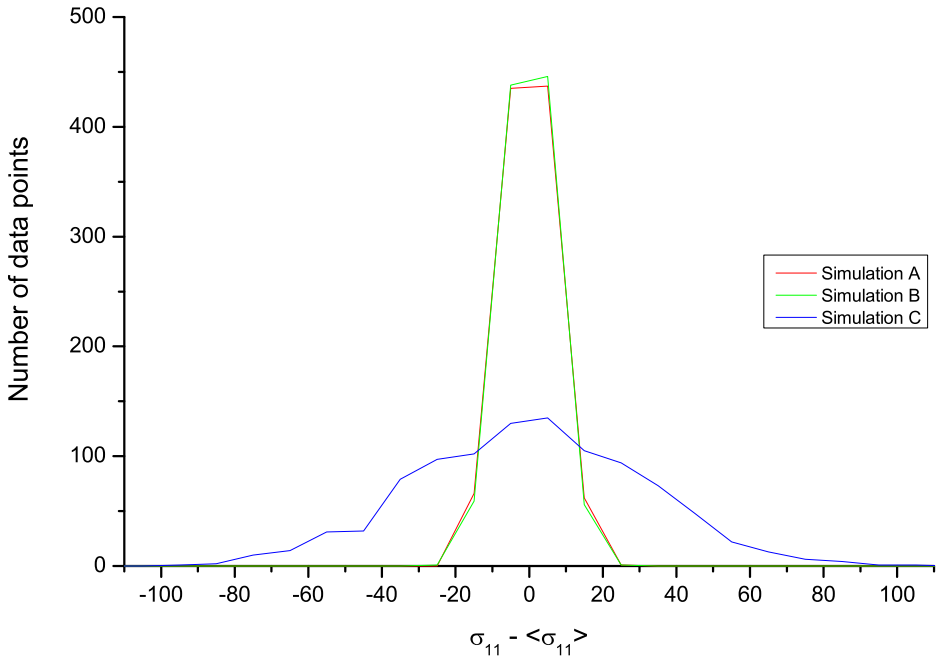


Figure 3.4: Histogram of the deviations from the average value of σ_{11} along the plateau of plots in Figure 3.3.

3.3.3 Choice of δ^* and its influence on simulated polymer stresses

To ascertain the effects of the extensibility parameter, δ^*, on simulated values of first normal polymer stress coefficient, Hsieh et al. [48] ran simulations using the same parameters as for their studies into the effects of changing the spring constant H^* used, with a few changes:

- Values of δ^* used are 0.05, 0.01 and 0.001.
- $H^* = 10000$ and $\Delta t^* = 0.0001$.
- The full run time of the simulation is 2.5 time units.

Again, using Equation (3.3.4), the parameters to be used for our simulations were calculated and are given in Table 3.6. No other parameter changes were made from those used for the simulations investigating the effects of the spring constant H^*. Figure 3.5 displays the plots of the first normal polymer stress coefficient, σ_{11}, against time for the three different extensibility parameter values (see Table 3.6) and a black line representing the data from [48, Figure 4] (extracted using Plot Digitizer). These plots indicate that

Table 3.6: Values of various parameters under the non-dimensionalisation paradigm of this work converted from the parameters used in [48] to analyse the effect of the choice of spring extensibility parameter on simulation results.

Simulation	Δt^*	Run Time	Q_0^*	δ^*	$\dot{g}^*$	δ^*
A				5		0.05
B	4	10^5	100	1	1.455×10^{-4}	0.01
C				0.1		0.001

the use of a small value for the extensibility $\delta^* = 0.001$ (simulation C in our work, with $\delta^* = 0.1$) results in large fluctuations in the σ_{11} value over time, compared with the use of extensibility values an order of magnitude greater (simulations A and B). This pattern was observed both in the data reported by Hsieh et al. [48] and by the simulation data documented here. This pattern is also observed in the study of the spread of σ_{11} away from the average along the plateau (defined here to be between times $t = 2.4$ and $t = 2.5$ inclusive), shown in Figure 3.6. The range is around four times larger for $\delta^* = 0.001$ than for either $\delta^* = 0.01$ or 0.05.

Table 3.7 lists data for the average values across the plateau of all time points between $t = 2.4$ and $t = 2.5$ of the mean value of σ_{11} and corresponding standard error at each time point. This data, along with magnified plots of the plateau shown in Figure 3.7, indicate the trend for the average value of σ_{11} to decrease as the extensibility parameter value decreases. This makes sense as, the lower the extensiblity parameter for a spring, the more the spring behaves like a rigid rod and the less likely a spring will stretch to lengths that give rise to greater tension forces.

However, this data also indicates that, as the extensibility parameter value decrease, the noise becomes more predominant faster in our simulations than in the data published by Hsieh et al. [48]. There are also discrepancies between the average plateau values for σ_{11} between the two sets of data.

Table 3.7: Averaged values for the mean and standard error of σ_{11} between time-points 2.4 and 2.5 (the "plateau") in simulations using parameters described in Table 3.6 and in [48, Figure 4].

Hsieh data mean	Simulation results		Simulation	δ^*
	Mean	Standard Error		
626.5	629.5	2.2	A	0.05
611.4	623.7	4.0	B	0.01
604.4	568.8	29	C	0.001

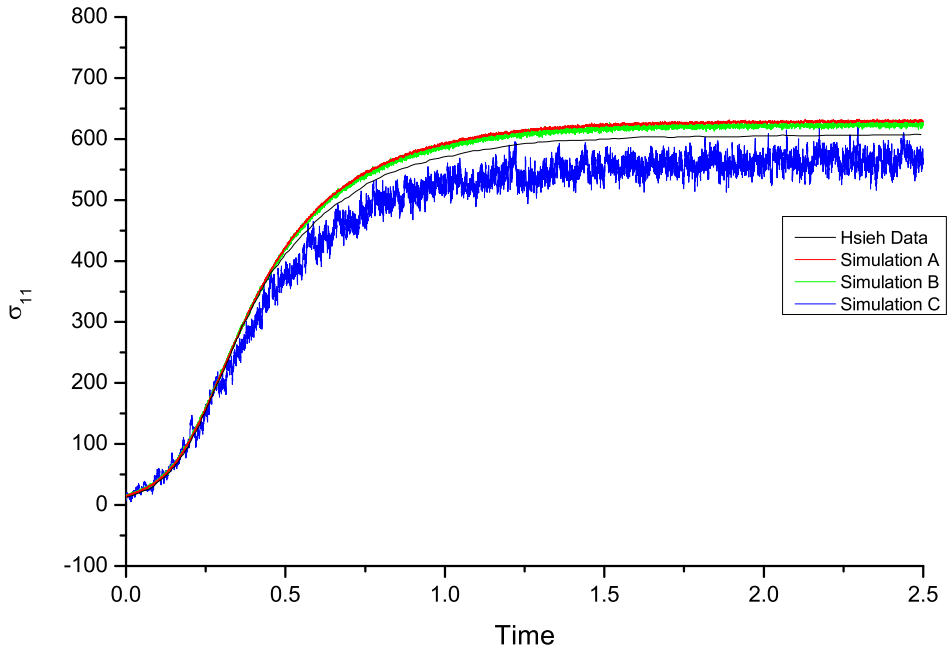


Figure 3.5: Plots of σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow with different spring extensibility values, as given in Table 3.6. Time scales of the simulations have been rescaled to match that of the data given in Hsieh et al. [48]. The "Hsieh data" plot corresponds to data obtained from Figure 4 in Hsieh et al. [48]

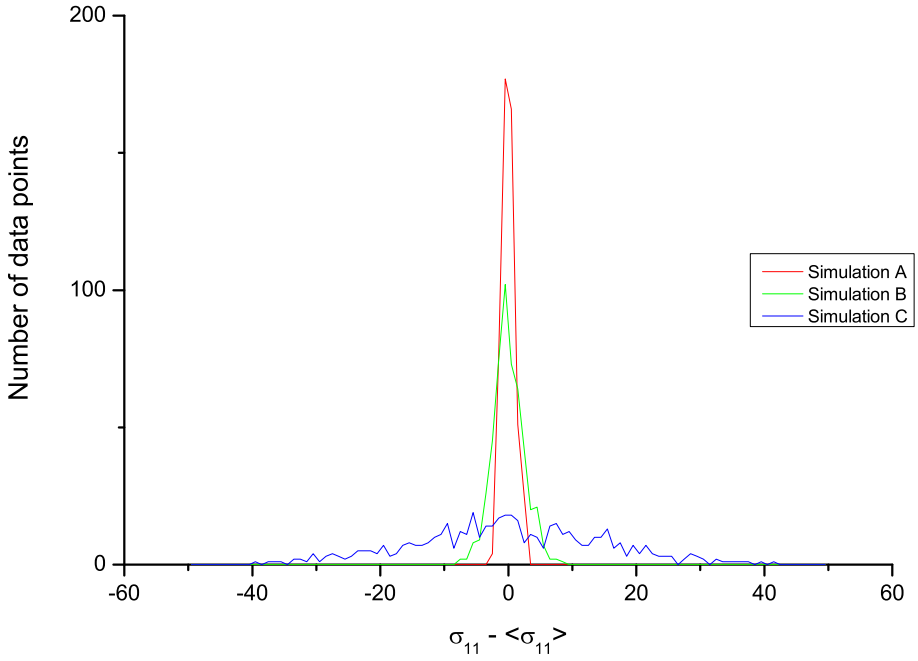
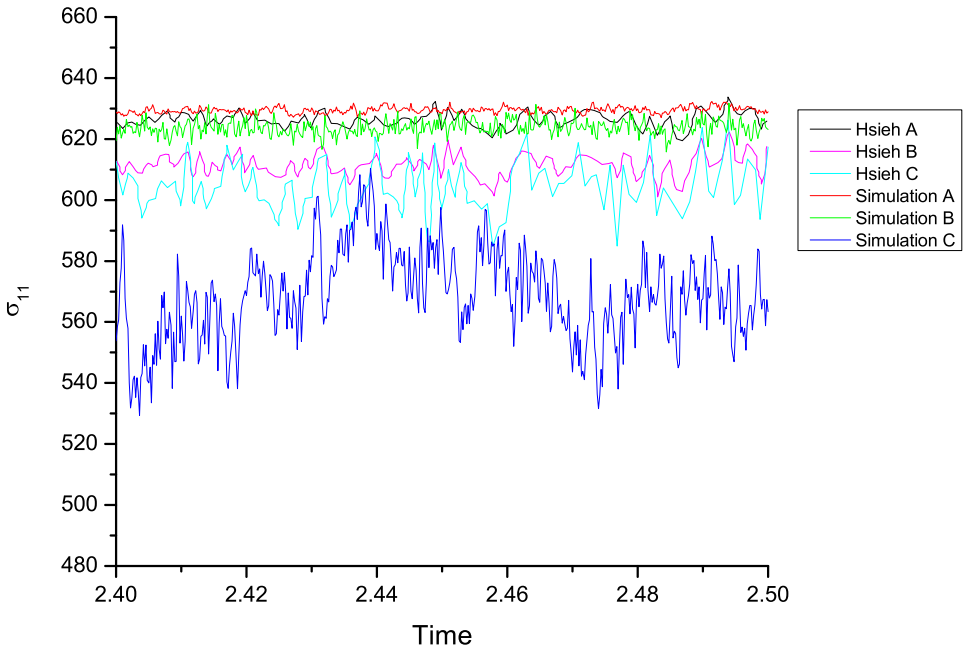
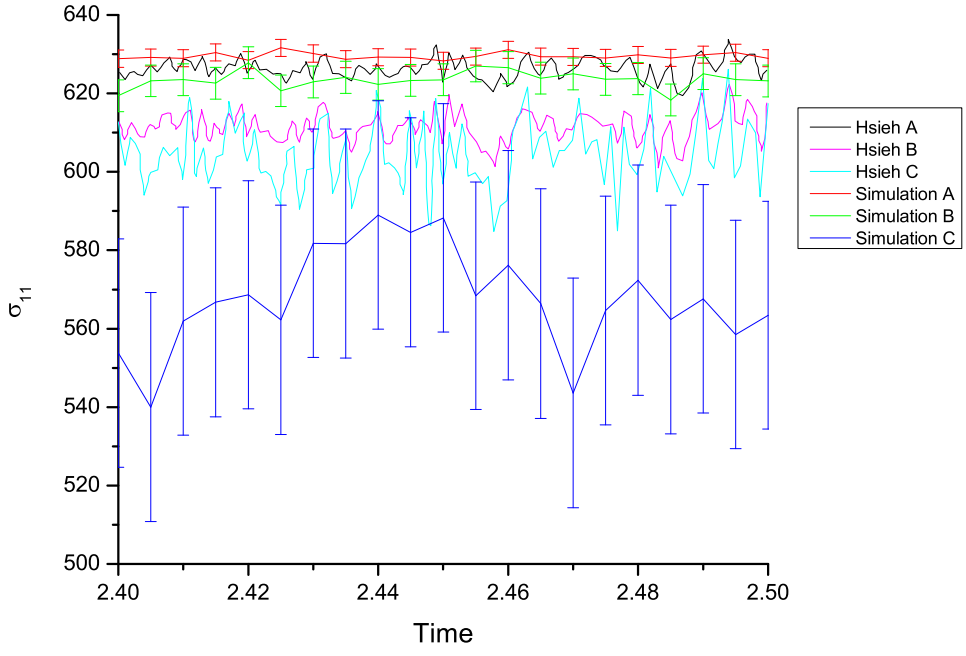


Figure 3.6: Histogram of the deviations from the average value of σ_{11} along the plateau of plots in Figure 3.5.



(a) Raw plots of the plateau data



(b) Reduced plot of simulation data with standard error bars

Figure 3.7: Magnification of the plots of Figure 3.5 in the plateau region between 2.4 and 2.5 time units. The lower plot features simulation data sampled at fewer time points, but includes the standard error bars at those points. These are based on a sample size of 10000 independent trajectories. Hsieh data obtained from [48, Figure 4 Inset] using Plot Digitizer.

3.3.4 Choice of Δt^* and its influence on simulated polymer stresses

To investigate the effects of the time-step size, Δt^*, on simulated values of first normal polymer stress coefficient, the parameters used in [48, §III. A.] were as for the investigation into the effect of the spring constant value, H^*, as previously discussed, but with a few changes:

- Values of Δt^* used are 0.001 and 0.005.
- $H^* = 10000$ and $\delta^* = 0.01$.
- Results are averaged over 1000 individual, independent trajectories. (Not 10000 as for previous simulations.)
- The full run time of the simulation is 4 time units.

Once again, using Equation (3.3.4), these parameters were converted into the parameter values given in Table 3.8 for use in our simulations. Furthermore, results are averaged over 1000 individual, independent trajectories; this was done to match the performance of our simulations against those documented in [48].

Table 3.8: Values of various parameters under the non-dimensionalisation paradigm of this work converted from the parameters used in [48] to analyse the effect of the time-step size on simulation results.

Simulation	Δt^*	Run Time	Q_0^*	δ^*	$\dot{g}^*$	Δt^*
A	40	1.6×10^5	100	1	1.455×10^{-4}	1×10^{-3}
B	200					5×10^{-3}

Figure 3.8 displays the plots of the first normal polymer stress coefficient, σ_{11}, against time for the two choices of time-step size (see Table 3.8) and a black line representing the data from [48, Figure 7] (extracted using Plot Digitizer). These plots initially suggest that the output values in our simulations are no further than 10% away from the data published by Hsieh et al. [48].

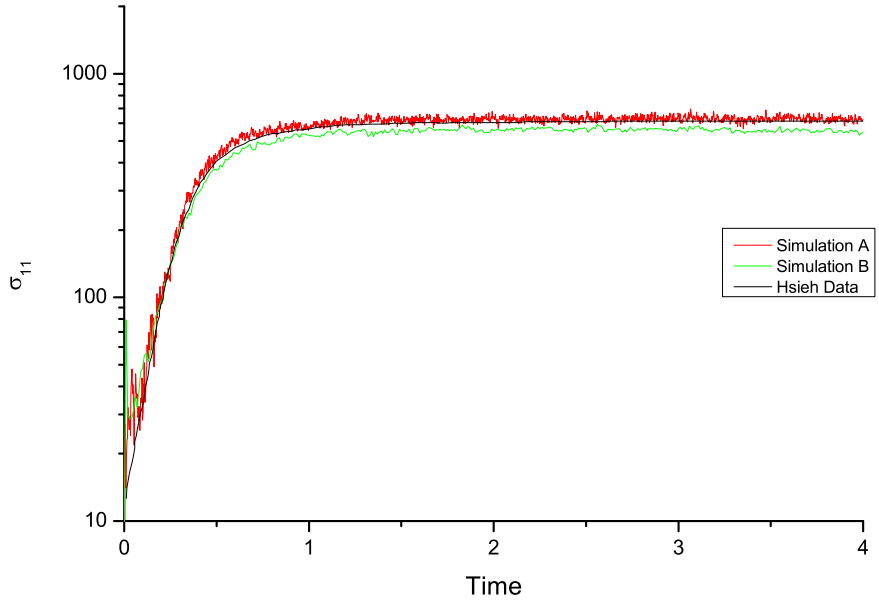


Figure 3.8: Plots of σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, corresponding to different time-step sizes, as described in Table 3.8. The "Hsieh data" plot corresponds to data obtained from [48, Figure 7]. Time scales of the simulations have been rescaled to match that of the data given in [48].

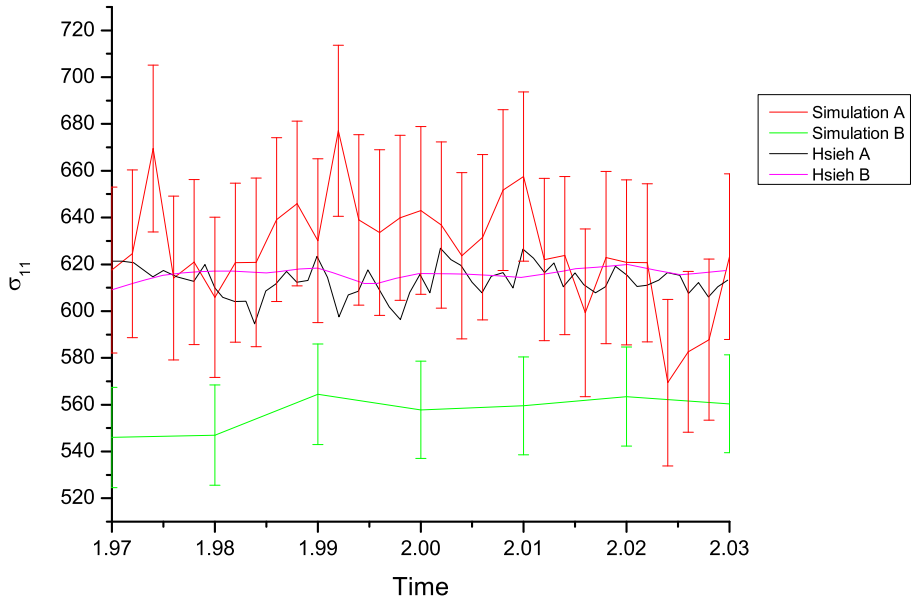


Figure 3.9: Magnification of the plots of Figure 3.5 in the plateau region between 1.97 and 2.03 time units, including standard error bars at each time point (i.e. Equation (3.2.1) is applied but with data with respect to a single time point), for our simulations based on a sample size of 1000 independent trajectories. Hsieh data obtained from [48, Figure 7 Inset] using Plot Digitizer.

Figure 3.9 is a magnification of a section of the plateau between time points $t = 1.97$ and $t = 2.03$, and includes standard error bars at individual time points for our simulation data. This figure shows that greater fluctuations were observed with the use of a smaller time step in both our simulation data and the Hsieh data; this makes sense as the smaller time step means a more refined simulation. However, the use of a large time step in our code (simulation B) results in a significant change in the value of σ_{11} observed. This is not seen in the data from [48, Figure 7 inset].

To ascertain the effect of the time-step size on simulations performed with our code, a number of other values of Δt^*, given in Table 3.9, were chosen with all other parameters left unchanged. As per Figure 3.9, values between 1.97 and 2.03 time units were considered; these are displayed, with standard error bars for each time point, in Figure 3.10. The means and standard deviations of values of σ_{11} across all samples and time points in this range are given in Table 3.9. Plots of the data in this range are also given in Figure 3.10, with a selection of the data given with standard error bars at some time points displayed in Figure 3.11.

The data indicates that for values of Δt^* above 80, the outputs of σ_{11} are lower than 620, implied by the results published by Hsieh et al. [48] to be the correct value for the choice of parameter values. For smaller time-step sizes, the values of σ_{11} obtained are around 620. However, there are large fluctuations present; these are greater in size than those observed where a large time-step size has been used. Furthermore, the magnitude of the standard error increases with decreasing time-step size, as demonstrated in Table 3.9 and Figure 3.11. We can therefore conclude that, as the time-step increases, the accuracy of the simulation increases, but the precision decreases.

Table 3.9: Values of Δt^* tested to ascertain the effect of time-step size on our simulations, with values of the average mean and standard error for σ_{11} between 1.97 and 2.03 time units (as scaled according to the simulation results of Hsieh et al. [48]).

Δt^*	σ_{11} between 1.97 and 2.03 time units	
	Mean	Standard Error
5	625	40.0
10	616	38.2
20	624	37.0
40	626	35.3
80	608	29.2
120	582	24.8
160	557	22.1
200	557	21.2
240	556	20.3

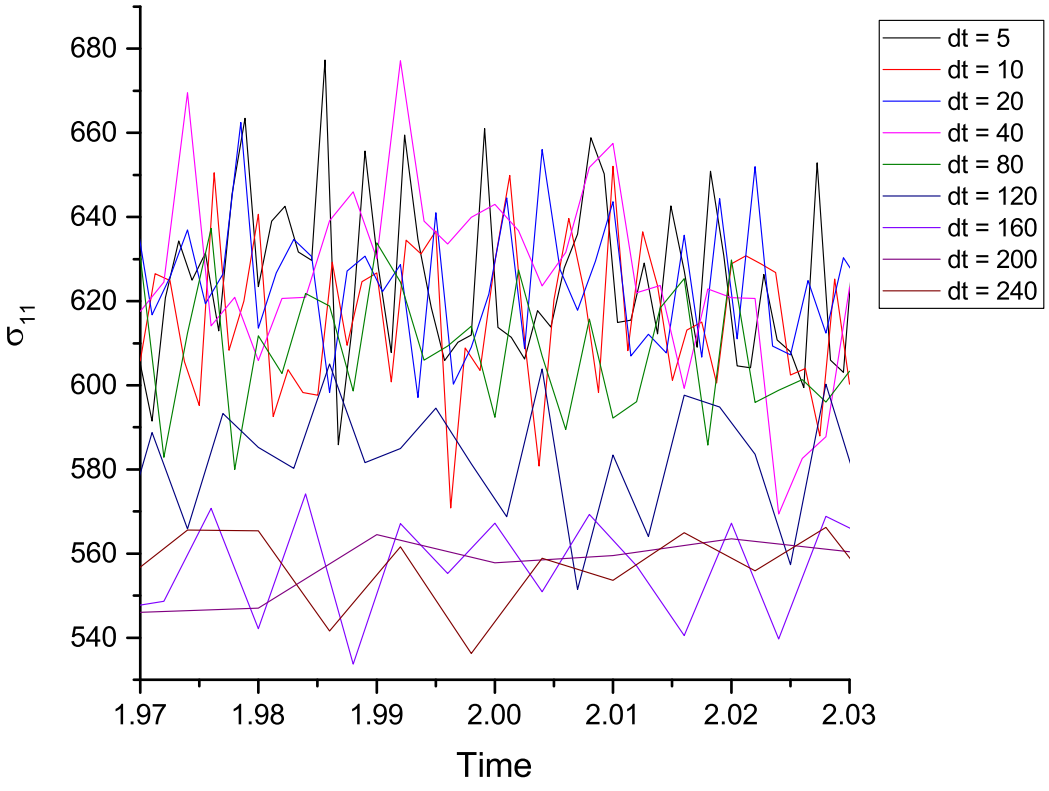


Figure 3.10: Plots of the σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, restricted to the plateau region between 1.97 and 2.03 time units (as scaled according to the simulation results of Hsieh et al. [48]) for a range of values of Δt^* as detailled in Table 3.9. The simulations were based on a sample size of 1000 independent trajectories.

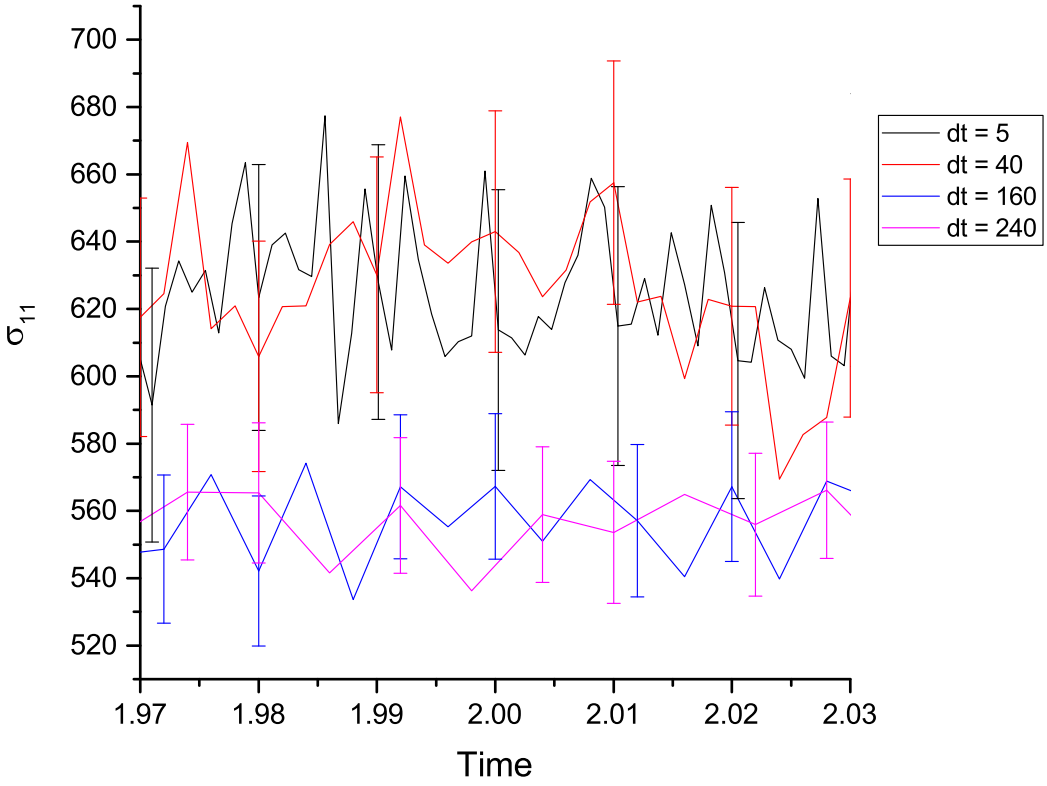


Figure 3.11: Plots of the σ_{11} against time for a 11-bead, bead-spring chain with FENE-Fr springs under a uniaxial extensional flow, as in Figure 3.10, for a selection of values of Δt^*. Standard error bars at a selection of time points are included (Equation (3.2.1) is applied but with data with respect to a single time point).

3.3.5 Summary of comparisons with data from Hsieh et al. [48]

Across the tests performed, the simulations carried out in this work have produced values that closely resemble the values reported by Hsieh et al. [48], with similar patterns observed in the effect of parameter changes to the noise of simulations, such as the increase in noise due to increased stiffness in the springs from a low extensibility parameter value. This suggests that the code is producing sensible outputs.

There are discrepancies between the values obtained through simulations in this work and the values reported in [48], though, with a sutiable choice of parameter values (particularly a small time-step size), these differences can be reduced to the order of 1% of the values obtained by Hsieh et al. [48]. However, the scale of noise in the simulation plots is generally larger than in the data published by Hsieh et al. [48]. Although a common algorithm is used in both works, it is possible that slight differences in the coding may result in the reduced precision in the outputs here. Changes were made to the original code as used by [117] to accommodate the extra precision required to run calculations with the FENE-Fraenkel spring force law. It is possible that the larger noise inherent in the results published here is due to the enhanced sensitivity to small changes that this increased precision in stored values allows for.

3.4 Orientation behaviour of a FENE-Fraenkel dumbbell in linear shear flows with and without hydrodynamic interactions

Since the main investigation is focussed on the orientation of molecules in fluid flows to give a linear dichroism signal, it was decided to try to use the simulation program to produce outputs of the behaviour of bead-spring chains in flow. To simulate polymer molecules, such as DNA, chains involving multiple beads and springs would normally be used. However, due to time limits on the use of computers to run the simulations, only dumbbells (*i.e.* a chain with 2 beads and 1 spring) were simulated. Running 100000 samples with dumbbells would typically last just under 2 days, the maximum time allowed for a process on the supercomputers used for this project. The use of longer chains would fail in the time limits given. While it was possible to split the simulations into smaller batches with a subset of the total number of runs, the availability of nodes on the supercomputer to run these jobs became a limiting factor.

The code had previously been used in the production of simulation data published by Pham et al. [108]. This data related to the behaviour of FENE springs, rather than the FENE-Fraenkel springs studied here. It was decided, therefore, to use parameter values chosen for comparison against results published by Hsieh et al. [48] as a basis for simulations of a dumbbell in a fluid shear flow. Table 3.10 details the choice of parameter values for these simulations.

Table 3.10: Parameters used for simulations to assess the alignment of FENE-Fraenkel dumbbells in shear flow.

Variable	Inputs
Number of beads, N	2
Spring natural length, Q_0	100
Spring extensibility, δ	1
Linear shear strain rates, $\dot{g}$ / 10^{-6}	5, 10, 50, 100
Hydrodynamic interaction parameter, h	28.2112421883
Tolerance level	1×10^{-8}
Time-step size, Δt	0.5, 1, 2, 4
Run time	100000
Number of samples	100000

The spring natural length and extensibility values, $Q_0^* = 100$ and $\delta^* = 1$, were used in simulations to compare against the work of Hsieh et al. [48], and therefore these values were deemed the best starting point for parameter values here. The results of the previous section also demonstrated that the use of small time steps of the order of $\Delta t^* = 4$ would mean that simulations would be refined (*i.e.* display details of the behaviour of a spring) without severely affecting the speed of the simulation. Larger values were shown to cause major changes in the simulation results (Figure 3.9).

The extensional strain rate value of 1.455×10^{-4} was used in most of the comparisons against data from [48]. In these simulations, a linear shear rate was used to try to mimic the environment of Couette flow. Strictly speaking, Couette shear flow is non-linear, but can be closely approximated by a linear shear flow depending on the ratio of the radii of the inner and outer cylinders. For example, as explained in Section 1.1, the Couette shear flow of the experimental setup detailled in this thesis can be approximated by a linear shear flow.

The upper value for the shear strain rate of 1×10^{-4} was chosen, as it was thought that a similar magnitude of shear strain would cause similar molecule stretching behaviour to that seen with extensional flow in simulations to compare against [48]. Lower values of 5×10^{-5}, 1×10^{-5} and 5×10^{-6} were chosen to provide an assessment of how the behaviour of a FENE-Fraenkel spring would be affected by lower strain rates.

Recall from Subsection 2.7.1 above that hydrodynamic interactions (HI) are encoded into the simulation through a non-dimensional parameter, h^*. In the case where HI are not considered, this parameter is set to zero. To derive a suitable non-zero value for the case when HI is included, we refer to Pham et al. [108]. This paper explains that the dimensional form of the hydrodynamic interaction parameter, h, is related to a non-dimensional form depending on the type of model being implemented.

For bead-rod models:

$$h = h_k^* \sqrt{N_k + 1}$$

with

$$h_k^* = \sqrt{\frac{3}{\pi}} \bar{R}_{hyd}$$

where:

- N_k is the number of *Kuhn steps*. A Kuhn step or segment is a section of a polymer chain considered to be freely jointed with other sections of the chain; that is, the orientation of a Kuhn step is independent of the orientation of surrounding sections of the chain.
- $\bar{R}_{hyd}$ is the dimensionless bead radius.
- h_k^* is the dimensionless hydrodynamic interaction parameter for bead-rod models.

For bead-spring models:

$$h = \tilde{h}^* \sqrt{N}$$

with

$$\tilde{h}^* = \frac{h^*}{\chi} = h^* \times \sqrt{\frac{3}{\langle Q^2 \rangle}}$$

where:

- N is the number of beads in the chain.
- $\langle Q^2 \rangle$ is the non-dimensionalised equilibrium average of the second moment of a spring in the chain.
- h^* is the hydrodynamic interaction parameter for a freely-jointed (Rouse [124]) bead-spring chain. This is the parameter for which a value is required for our simulations.

In their work, Pham et al. [108] use a variety of different configurations to relate the hydrodynamic interactions of a bead-rod chain to that of a bead-spring chain — in particular, the choice of value for $\tilde{h}^*$ with respect to h_k^*, so that the hydrodynamic interactions between the two schemes can be compared as the number of springs in a chain is increased.

In this work, consideration is given only to the simple case of $\tilde{h}^* = h_k^*$, which corresponds to situation where the dimensional HI parameter h is the same for both the bead-rod and bead-spring configurations as number of springs in the bead-spring chain is increased to the number of Kuhn steps in the bead-rod chain. The consequences of this situation and other alternatives are discussed by Pham et al. [108], particularly with regard to cases where the addition of too many springs to a bead-spring chain has a detrimental effect. However, for the simulation work here, this case is made only to provide a test value to use for h^* in a basic 2-bead system.

As a starting point, we shall use the values used by Pham et al. [108] — $\bar{R}_{hyd} = 0.5$ and $h_k^* = \sqrt{3/\pi}\bar{R}_{hyd}$. Therefore:

$$\tilde{h}^* = h_k^* = \frac{1}{2}\sqrt{\frac{3}{\pi}},$$

and

$$h^* = \tilde{h}^* \sqrt{\frac{\langle Q^2 \rangle}{3}} = \frac{1}{2}\sqrt{\frac{3}{\pi}} \times \sqrt{\frac{\langle Q^2 \rangle}{3}} = \frac{1}{2}\sqrt{\frac{\langle Q^2 \rangle}{\pi}}$$

Since FENE-Fraenkel springs with natural length $Q_0 = 100$ and maximum extension $\delta = 1$ (therefore $q_0 = Q_0/\delta = 100$) are being implemented, Equation (2.4.13) can be used to calculate $\langle Q^2 \rangle$.

$$\langle Q^2 \rangle = \frac{\left[\left(\frac{3}{6} + 60000\right)\left(\frac{1}{4}\right) + 100000000\right]}{\left(\frac{1}{4} + 10000\right)} = \frac{100015000.125}{10000.25}$$

Therefore

$$h^* = \frac{1}{2}\sqrt{\frac{\langle Q^2 \rangle}{\pi}} = 28.211$$

The tolerance refers to a part of the algorithm for calculating spring lengths at each time step, as describe in Subsection 2.7.4. The algorithm proceeds to the next time-point only when the change in successive iterates of the spring length value at the current time-

point is below a certain tolerance threshold. The algorithm described in this work was also implemented by Hsieh et al. [48] and their results use a tolerance of 10^{-8}. Smaller tolerances were tested and deemed not to result in any noticeable change in the data output (values did not change by more than 0.1%), though the simulation times were larger.

While previous simulations were run with 10000, the increased noise in the data outputs from our code compared with the data documented in [48] suggested that the number of trajectories (*i.e.* the sample size) should be increased by at least one order of magnitude. A sample size of 100000 was chosen as it provided the largest number of trajectories to be simulated, without leading to simulations running for too long.

For molecules in a sampled aligned uniaxially, recall from the introduction that the orientation parameter, S, a value expressing the proportion of molecules aligned along an orientation axis, can be expressed as:

$$S = \frac{1}{2}\left(3\langle\cos^2\theta\rangle - 1\right)$$

where θ is the angle between the axis of a molecule and the orientation axis, and $\langle\cos^2\theta\rangle$ is averaged across all the molecules in a sample. It is this parameter value that was measured and is reported in the data below.

This calculation was implemented using the following code. Note that these lines of code replace the lines corresponding to property 9 in `\properties.f90` (see Appendices B.5 and D.5 for further details).

```
props(9) = 0.0
Do mu = 2, NBeads
   interimlength = sum((R(:,mu) - R(:,mu-1))*(R(:,mu) - R(:,mu-1)))
   interimlength1 = (R(1,mu) - R(1,mu-1))*(R(1,mu) - R(1,mu-1))
   props(9) = props(9) + (interimlength1/interimlength)
end Do
props(9) = (1.5d0*(props(9)/(NBeads-1))) - 0.5d0
```

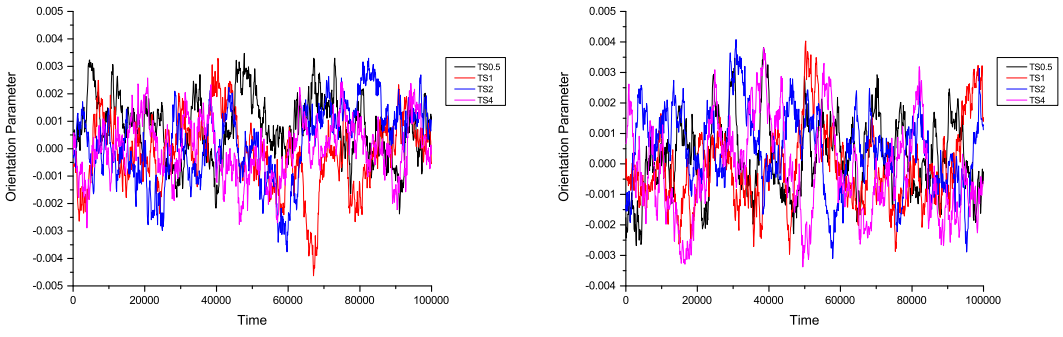
Here `R` corresponds to a matrix containing the coordinates of the bead positions, with `mu` acting as the variable for the bead number. For each spring, the code above calculates the square cosine of the angle between the alignment axis (taken to be the "x-axis", *i.e.* along

the line of the first coordinate position) and the molecule. This is done by squaring the length of the spring in the first coordinate direction and dividing by the squared length of the spring. In the third-last and last lines, the values of the squared cosines across all springs in a bead-spring chain are averaged and this is used in the final calculation of the chain's orientation parameter. The motivation for this is related to the way that LD signals from polymers are gathered. Chromophores (and their orientation) in sections of the molecule give rise to an LD signal; thus each spring of the chain is treated as possessing chromophores that are oriented the same way and independently of chromorphores in another spring or section of the chain (akin to the idea of the Kuhn step in bead-rod chains).

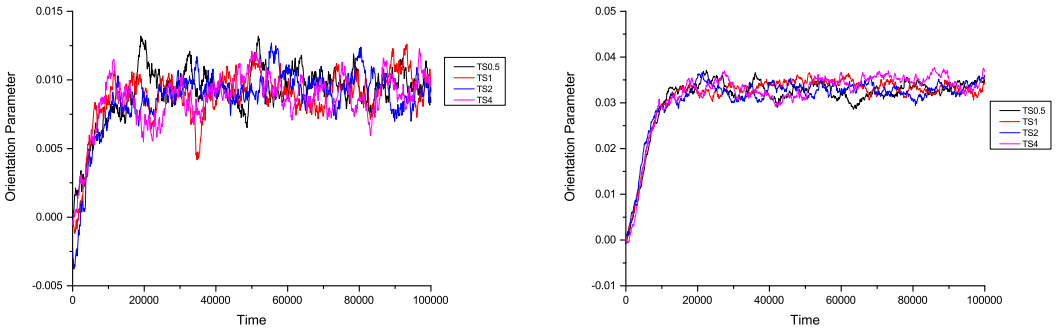
Table 3.11: Mean values and standard errors for the orientation parameter S (to 3 decimal places) under different linear shear rates, different time-step sizes and with or without hydrodynamic interactions present. These have been averaged over the timepoints of the last 20000 time units of the 100000 time unit length simulation.

Shear rate $\dot{g}$	Without hydrodynamic interactions		With hydrodynamic interactions		Time-step size
	Mean	Standard Error	Mean	Standard Error	Δt
5×10^{-6}	1.8×10^{-4}	2.8×10^{-3}	-4.4×10^{-4}	2.8×10^{-3}	0.5
	3.0×10^{-4}	2.8×10^{-3}	6.0×10^{-4}	2.8×10^{-3}	1
	1.5×10^{-3}	2.8×10^{-3}	1.0×10^{-3}	2.8×10^{-3}	2
	4.9×10^{-5}	2.8×10^{-3}	4.3×10^{-4}	2.8×10^{-3}	4
1×10^{-5}	2.8×10^{-5}	2.8×10^{-3}	-5.5×10^{-4}	2.8×10^{-3}	0.5
	4.4×10^{-4}	2.8×10^{-3}	1.8×10^{-3}	2.8×10^{-3}	1
	1.4×10^{-5}	2.8×10^{-3}	-4.3×10^{-4}	2.8×10^{-3}	2
	-6.8×10^{-4}	2.8×10^{-3}	-7.9×10^{-4}	2.8×10^{-3}	4
5×10^{-5}	9.5×10^{-3}	2.8×10^{-3}	1.1×10^{-2}	2.8×10^{-3}	0.5
	9.8×10^{-3}	2.8×10^{-3}	9.8×10^{-3}	2.8×10^{-3}	1
	9.3×10^{-3}	2.8×10^{-3}	8.5×10^{-3}	2.8×10^{-3}	2
	9.2×10^{-3}	2.8×10^{-3}	9.3×10^{-3}	2.8×10^{-3}	4
1×10^{-4}	3.4×10^{-2}	2.9×10^{-3}	3.5×10^{-2}	2.9×10^{-3}	0.5
	3.3×10^{-2}	2.9×10^{-3}	3.1×10^{-2}	2.9×10^{-3}	1
	3.3×10^{-2}	2.9×10^{-3}	3.3×10^{-2}	2.9×10^{-3}	2
	3.5×10^{-2}	2.9×10^{-3}	3.5×10^{-2}	2.9×10^{-3}	4

Table 3.11 displays the orientation parameter mean and standard error (from a sample of 100000 trajectories) averaged across time points from the 20000 time units of the duration of the simulation. The reason for this choice is that any relatively large changes in the value of the orientation parameter do not appear to take place in this time interval. This can be seen in Figure 3.12 (in particular Figures 3.12c and 3.12d) and Figure 3.13 (in particular Figures 3.13c and 3.13d); these display the mean value of the orientation



(a) $\dot{g} = 5 \times 10^{-6}$ **(b)** $\dot{g} = 1 \times 10^{-5}$



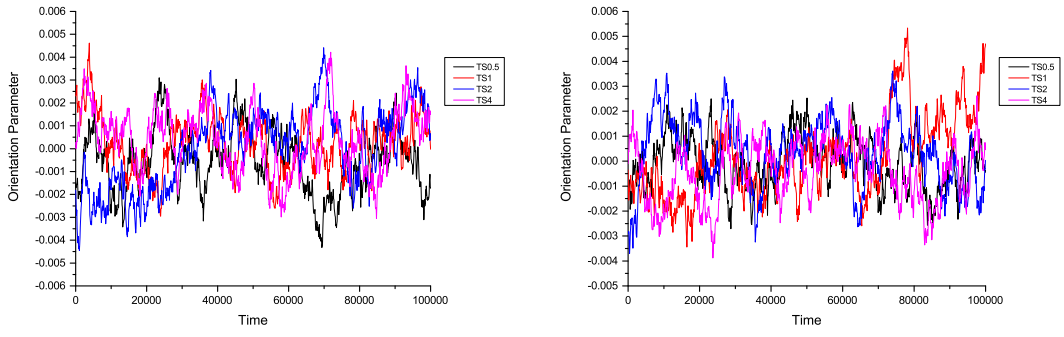
(c) $\dot{g} = 5 \times 10^{-5}$ **(d)** $\dot{g} = 1 \times 10^{-4}$

Figure 3.12: Plots of the orientation parameter S against time for different time-step values and different linear shear rates $\dot{g}$ without hydrodynamic interactions, using a FENE-Fraenkel spring with natural length $Q_0 = 100$ and extensibility $\delta = 1$. Standard error bars have been ommitted for clarity.

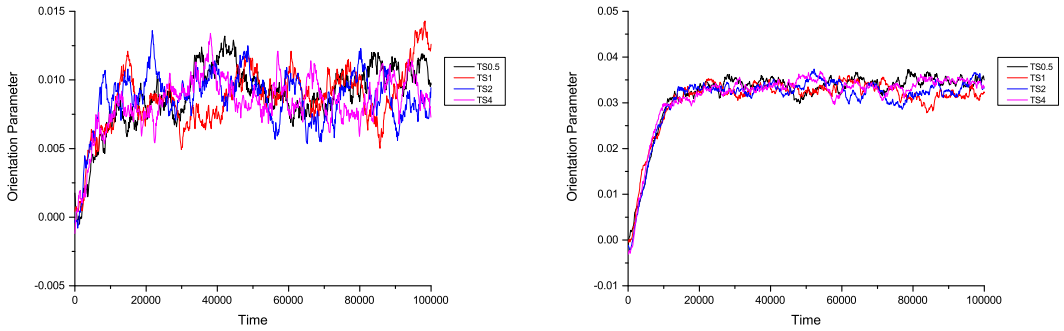
parameter with and without HI and using different shear strain rates and time-step sizes.

For strain rates $\dot{g} = 5 \times 10^{-6}$ and $\dot{g} = 1 \times 10^{-5}$, there is no consistent mean value for the orientation parameter. Figures 3.12a, 3.12b, 3.13a and 3.13b display a large level of noise in the behaviour of the mean orientation parameter against time. Furthermore, the oscilations of the mean value are also well within the standard errors tabulated. This information indicates that these strain rates are too low for the orientation parameter to settle on a value within 100000 time units.

However, with $\dot{g} = 5 \times 10^{-5}$ and $\dot{g} = 1 \times 10^{-4}$, across all time-step value choices, there is more consistency in the mean orientation parameter value. This is also borne out in plots shown in Figures 3.12c, 3.12d, 3.13c and 3.13d. For $\dot{g} = 5 \times 10^{-5}$, the mean values across all choices of time-step value are around 9.5×10^{-3}, though there is still a significant amount of noise in the plateau, with oscilations around the same magnitude as the standard error. However, when $\dot{g} = 1 \times 10^{-4}$, a higher mean value of the orientation



(a) $\dot{g} = 5 \times 10^{-6}$ **(b)** $\dot{g} = 1 \times 10^{-5}$

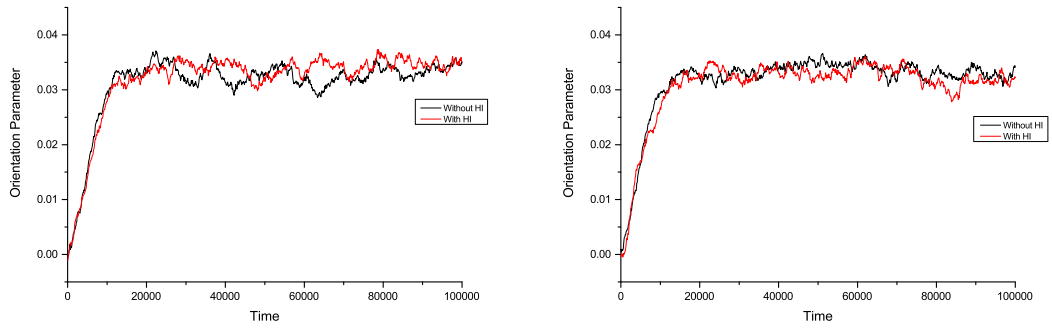


(c) $\dot{g} = 5 \times 10^{-5}$ **(d)** $\dot{g} = 1 \times 10^{-4}$

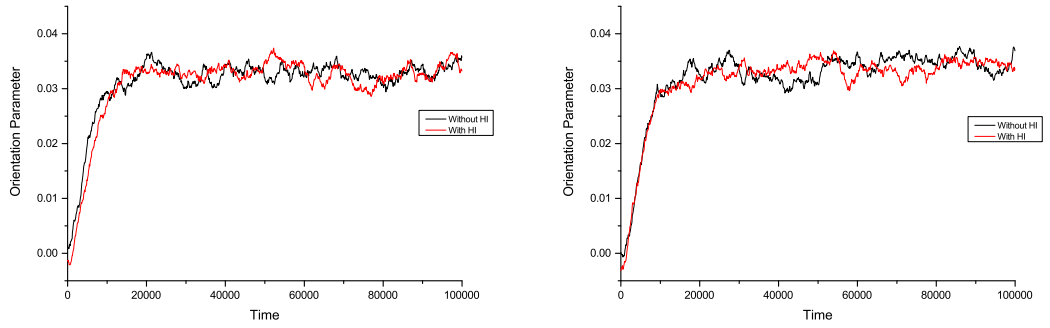
Figure 3.13: Plots of the orientation parameter S against time for different time-step values and different linear shear rates $\dot{g}$ with hydrodynamic interactions ($h^* = 28.211$), using a FENE-Fraenkel spring with natural length $Q_0 = 100$ and extensibility $\delta = 1$. Standard error bars have been ommitted for clarity.

parameter is observed (around 3.3×10^{-2}) with more consistency between the averages from simulations using different time-step sizes.

Figure 3.14 shows plots of the mean orientation parameter value against time for shear rate $\dot{g} = 1 \times 10^{-4}$ using different time-step sizes and compares the results with and without HI in the model. The data from simulations using lower shear rates has not been plotted as the plateau in the plots of S against time is the most discernable for this value of $\dot{g}$. The data indicates that there is no discernable change in the orientational behaviour of a FENE-Fraenkel dumbbell in this shear flow between the case when hydrodynamic interactions are included and when they are not. But as hydrodynamic interactions are intramolecular interactions, this result makes sense. Further simulations would need to be carried out with more beads and springs in the chain as a longer molecule would be subject to more hydrodynamic interactions than a shorter molecule. These tests would also be more comparable to the behaviour of polymers such as DNA.



(a) $\Delta t = 0.5$ **(b)** $\Delta t = 1$



(c) $\Delta t = 2$ **(d)** $\Delta t = 4$

Figure 3.14: Plots of the orientation parameter S against time with a shear rate value of $\dot{g} = 1 \times 10^{-4}$ for different time-step values with and without hydrodynamic interactions, using a FENE-Fraenkel spring with natural length $Q_0 = 100$ and extensibility $\delta = 1$. Standard error bars have been ommitted for clarity.

The simulation can also be refined with the inclusion of *bending potentials*, which apply an energy cost when adjacent springs connected to the same bead get too close to each other [46], and *excluded volume interactions* which have been discussed previously (Section 2.5). A further extension can be made by considering intermolecular interactions; that is, the interactions of chains with each other. The code used in this work is setup with chains in the so-called "dilute limit", where such interactions are not considered.

A previous study by McLachlan et al. [89] used a bead-rod model with two beads only to investigate the orientation of long rigid molecules, such as M13 bacteriophage particles. Figure 3.16 displays plots from those simulations. The simulations results shown in Figure 3.15 are similar to those obtained by McLachlan et al. [89] in two ways. First, the behaviour of the orientation parameter over time, with a consistent rise in value early on in the simulation before a plateau for the remainder; second, higher shear rates giving rise to higher plateau values for the orientation parameter. The overshoot and under-

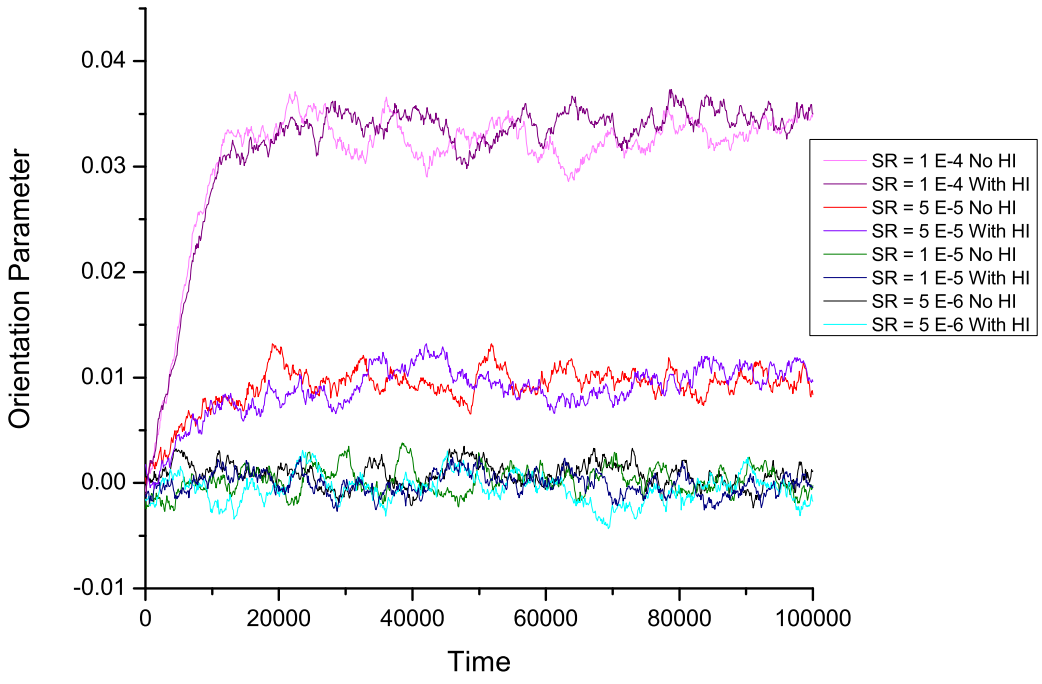


Figure 3.15: Plots of the orientation parameter S against time for all shear rate values tested (labelled SR in the diagram) with and without hydrodynamic interactions, using a FENE-Fraenkel spring with natural length $Q_0 = 100$, extensibility $\delta = 1$ and time-step size $\Delta t = 0.5$. Standard error bars have been ommitted for clarity.

shoot present in some of the plots in Figure 3.16, subfigure (b), is due to the means by which the orientation parameter is calculated; this is done by numerically solving the Fokker-Plank equation to obtain the orientation distribution function (a probability density function giving the probability of finding a rod at a particular angle). The method is described by Strand et al. [142] as giving rise to the "expected" overshoot phenomenon, as well as undershoot phenomena for systems with large Péclet numbers (i.e. where the advection forces are significantly greater than the diffusive forces acting on particles in a fluid). The Péclet number is roughly proportional to the Weissenberg number [137], which is the product of the shear rate and the stress relaxation time of the polymers in the fluid. It is therefore more likely that these overshoots and undershoots are observed with larger shear rates [137]. The presence of so-called "viscosity overshoots" in start-up flow has been studied experimentally [7, 101, 140]. Earlier work by Biller et al. [12] also indicates that the use of Hookean springs in simulations does not give rise to this overshoot phenomenon, whereas the use of FENE springs does.

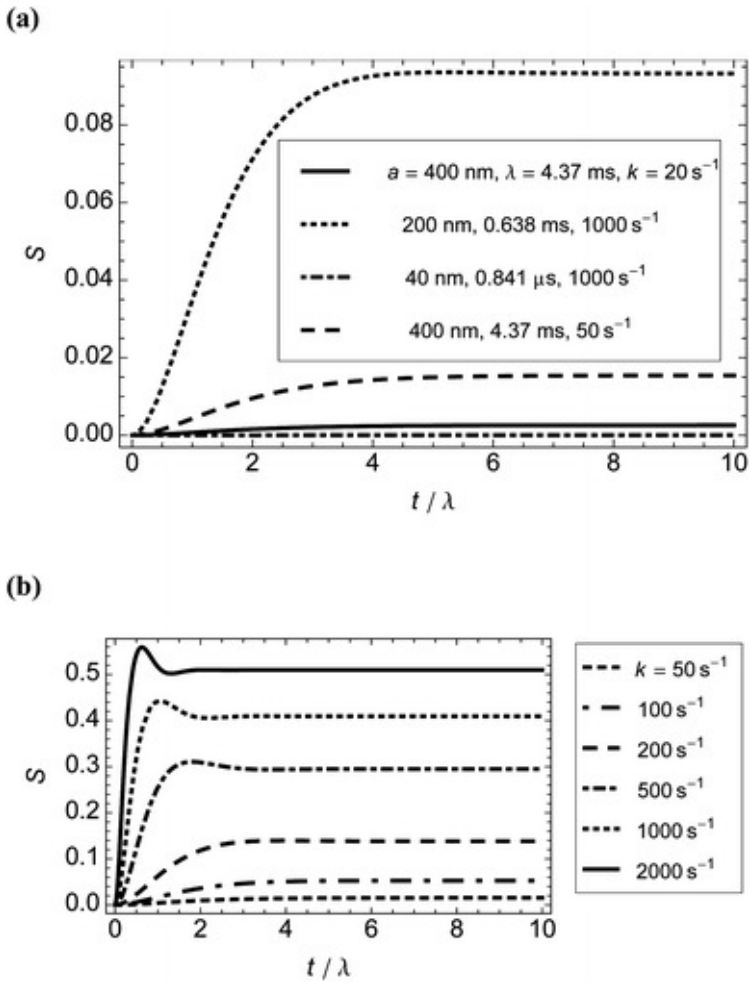


Figure 3.16: Plots of the orientation parameter, S, against time for different particle sizes and shear rates at 22°, as calculated from the model developed by McLachlan et al. [89]. a and b are the particle semi-diameters measured parallel and perpendicular, respectively, to the axis of revolution, k is the shear rate and λ is the particle material time constant (see [14]). Subfigure (a) concerns shorter particles ($b = 4$ nm) or low shear rates. Subfigure (b) focusses on high shear rates for M13 bacteriophage particles ($\lambda = 4.37$ ms). The graphs are reproduced from Figure 2 of [89].

One unusual pattern observed is the consistency of the size of the standard error, irrespective of the choice of shear strain rate, time-step size and presence of HI. It could be related to the spring parameters and the number of bead/spring in the chain, but further simulations would need to be performed with a variety of values for Q_0 and δ to ascertain if this link is valid.

A further cause for concern in the results is the presence of negative values for the orientation parameter. Theoretically, such values should not be possible since $0 \leq \langle \cos^2 \theta \rangle \leq 1$. Further testing of the code, perhaps isolating the calculation of the orientation parameter, would need to be carried out to establish is this error is systematic or a result of computational inaccuracies similar to the precision issues encountered when first attempting to

implement the FENE-Fraenkel force law into the code. It could also be an indication that the simplified formulation of the orientation parameter, as given in Equation (1.3.5), is not suitable here, and that Equation (1.3.4) should be used instead.

To draw further parallels between the simulations and experimental observations, the next step would be to use parameter values that are derived from real-world values for the length and stiffness of molecules.

3.5 Summary and future work

The FENE-Fraenkel spring force law has been implemented in code previously used to study bead-FENE-spring chains in flow. Changes were made to the code as it was found that the level of precision used by the code was insufficient to handle calculations with the FENE-Fraenkel spring force law. A further alteration was introduced to solve the cubic polynomial Equation (2.7.22) explicitly rather than with a Newton-Raphson iterative scheme, to remove the guesswork required in trialling a first guess for a root.

Comparisons with analytical zero-shear results indicate that the code behaves as the theory predicts. The data obtained is also comparable to the simulation work by Hsieh et al. [48], with similar patterns observed in the scale of "noise" of the output when parameters are changed to increase the spring stiffness. Small discrepancies with data in [48] do exist, but these are minimised with the a suitable choice of parameters, particularly with the use of a small time-step size. Changes to the code that were required to counteract issues caused by a lack of precision may be an underlying cause of the increased noise in our simulations compared with the data published by Hsieh et al. [48]. Simulations of the alignment of FENE-Fraenkel spring dumbbells in shear flows show increased alignment with high shear rates, similar to modelling data published by McLachlan et al. [89].

Since these simulations are for the study of polymers, further simulations should to be carried out with chains (*i.e.* more than 2 beads) in shear and extensional flow, to see how the alignment parameter changes over time in these flow regimes. These simulations should be performed with a wider range of strain rates, natural length and extensibility parameter values than used here. However, attention would need to be paid to how realistic the choice of parameters are, with respect to the behaviour of polymers in reality. In addition,

these simulations should also include excluded volume and hydrodynamic interactions, particularly as longer chains with more beads will increase the prevalence of these interactions. The code, as published here, is able to implement these interactions. Finally, an extension on the inclusion of excluded volume interactions would be the inclusion of bending potentials, as discussed by Holleran and Larson [46]; instead of a freely-jointed chain, there would be restrictions on the proximity of springs to each other, reflecting the finite flexibility of polymers in real life.

Chapter 4

Development of planar microfluidic devices for linear dichroism studies

DNA has been extensively studied with linear dichroism spectroscopy; in many cases, DNA molecules were aligned using Couette shear flow [97]. The behaviour of DNA molecules has also been studied in different flow regimes, using microfludic devices and microscopy. A number of these experiments have used extensional flow to control the motion of DNA molecules [135]. However there have not been any attempts to study DNA using linear dichroism in an extensional flow setting.

This chapter considers LD experiments performed to observe the alignment of DNA molecules in three flow regimes: Couette shear flow using existing equipment already developed in our lab, and pressure-driven (Poiseuille) and extensional flows using microfluidic devices manufactured from polydimethylsiloxane (PDMS). Preliminary experiments were performed using test designs to determine if LD experiments with planar microfluidic devices were feasible. The development of an extensional flow microfluidic device was considered, with further experiments performed to ascertain the ability of these devices to align DNA molecules for LD experiments in different types of fluid flow.

4.1 LD experiments using Couette Flow

LD experiments typically use a circular-dichroism spectropolarimeter with a rectangular beam of light passing through a sample chamber and then into a photomultiplier tube for detection. Two devices are used for LD experiments in this thesis — a Jasco J-815 CD spectropolarimeter and a BioLogic MOS-450 spectropolarimeter. Both devices use a Xenon arc lamp to generate a spectrum of light with wavelengths between 180 nm in the "near-UV" range and 1100 nm in the infra-red range. A polariser and a photoelastic modulator are used to generate two linear polarisations of the beam, such that they polarisations are perpendicular to one another.

For Couette flow LD experiments, a setup where a small volume, roughly 70 µl, of fluid sample is placed in a rotating quartz cuvette, with a stationary quartz rod placed in the centre (see Figure 4.1) was previously developed in our lab [82, 83]. The light beam passes through a focussing lens, through the cylindrical cuvette setup (see Figure 1.3) and then into a photomultiplier tube for detection.

The principle behind the use of Couette flow for LD experiments is that it a shear flow (see Figure 1.2). In particular, polymer molecules in shear flow align with the long axis parallel to the shear axis. So in this setup, the short axes of polymer molecules in a sample are aligned radially.

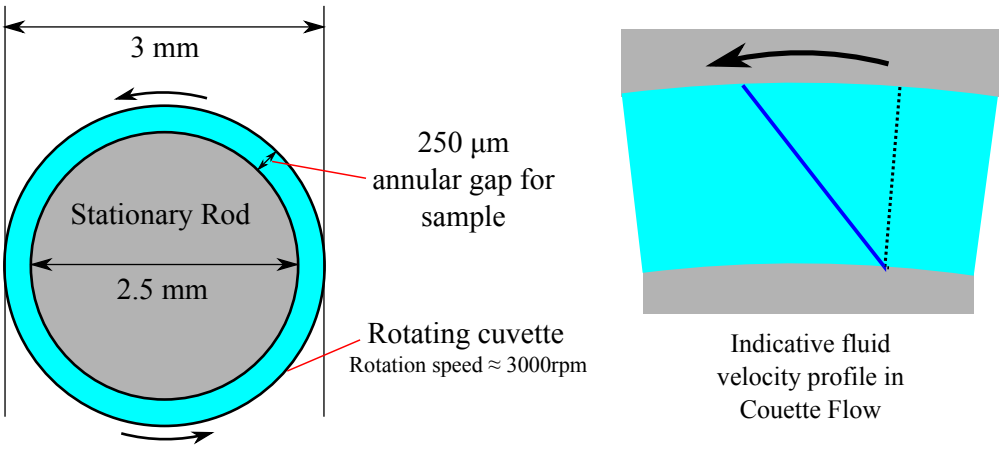


Figure 4.1: Schematic of the Couette flow cell used for LD experiments.

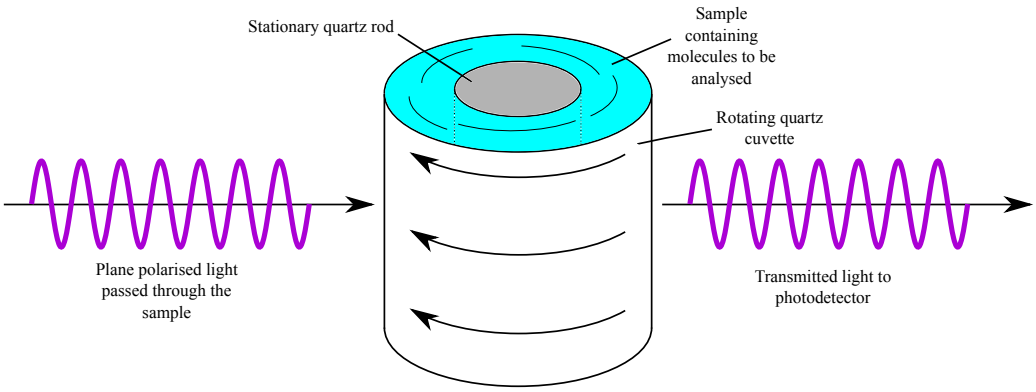


Figure 4.2: Schematic of a Couette flow cell for LD experiments.

4.2 Absorbance of PDMS

As previously mentioned, microfluidic "lab-on-a-chip" devices have also been used to study molecules and manipulate fluids to generate various patterns of fluid flow. A number of these devices have been constructed using polydimethyl siloxane (PDMS). Typically, the fluid channels in microfludic devices made from PDMS are on the scale of a few centimetres in length and width and between a few micrometers to one millimetre in depth [144]. Sylgard® 184 silcone elastomer, manufactured by Dow Corning, was used as the PDMS source for experiments in this thesis.

To ascertain if microfluidic devices made from PDMS can be used in LD experiments, the absorbance spectrum of the polymer was obtained to establish if it was transparent for various wavelengths of light. The results in Figure 4.3 show that PDMS is transparent down to 300 nm. Below this, PDMS begins to absorb in the near-UV region, with a significant increase in absorbance around 230 nm. DNA molecules are known to absorb around the 260 nm wavelength [97]. Therefore in developing further experiments with PDMS devices, care had to be taken to minimize the thickness of PDMS present in the light beam path to avoid absorbance from microfluidic device. Furthermore, the disc PDMS samples, formed from setting PDMS in 10 ml beakers, were formed with a curved, concave surface on one side, which may have resulted in light being scattered away from the detector

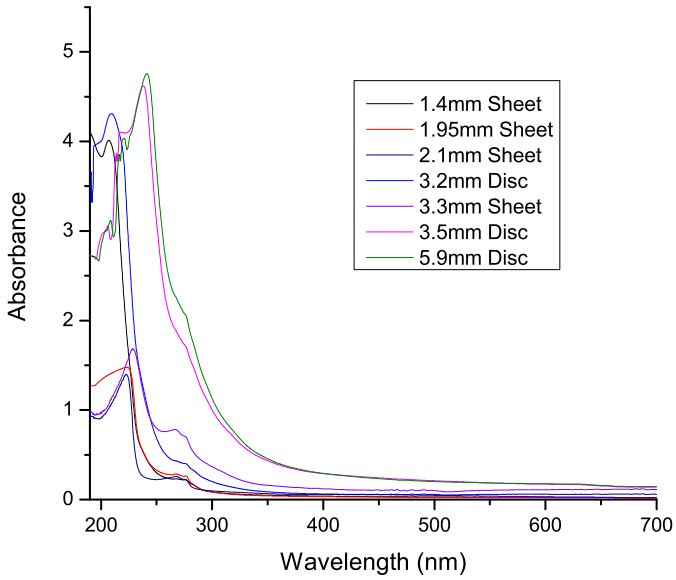


Figure 4.3: Absorbance spectra of polydimethylsiloxane (PDMS) samples with different thicknesses.

and higher absorbance readings. This demonstrated the importance of keeping the PDMS channel surfaces as flat as possible.

Since these measurements show that PDMS is transparent to unpolarised light above a certain wavelength, we can conclude that PDMS is transparent to polarised light in the same region. Therefore, given that samples being studied with PDMS microfluidic devices do not give an LD signal in this transparency region, a separate LD spectrum of PDMS was not obtained.

4.3 Preparation of planar microfluidic channels

The preparation of microfludic devices involve two phases: Making the silicon wafer cast, and then manufacture of the channel out of PDMS using the cast.

4.3.1 Making the silicon wafer cast

The manufacture of silicon wafer casts took place in a clean room to avoid dust and other debris from damaging the casts during the manufacture process. Channel designs (shown in sections below) were created in Inkscape, with each design prepared wholly in black on a white background. This allowed the designs to be printed in neagtive form on X-ray film (i.e. the channel design is a transparent region of the film).

For each cast, a clean, dust-free, dry silicon wafer was placed on a wafer spinner; care was taken to ensure that the wafer was balanced on and placed in the centre of the rotating holder, before turning on the vacuum to hold the wafer in place. Next, a quantity of SU-8 2100 photoresist liquid (manufactured by MicroChem) was carefully placed on the wafer centre in such a way as to avoid bubble formation. The photoresist is made up of an epoxy dissolved in an organic solvent. The specific quanitity is dependent upon the diameter of the wafer used — 1 ml of photoresist should be used for every 25 mm of wafer diameter. In the case of the experiments documented here, 10 cm-diameter wafers with 4 ml of photoresist were used.

After installing a cover inside the spinner lid to catch any residual photoresist spun off, each wafer was then spun to allow the photoresist to evenly coat the wafer surface. This step involves two substeps:

1. The wafer was spun at 500 rotations per minute (rpm) for 5-10 seconds with an acceleration of 100 rpm/s, to gently spread the liquid across as much of the wafer before the rotation speed increases in the second substep.

2. The wafer was then spun at a particular rotation speed between 1000 rpm and 3000 rpm for 30 seconds, with an acceleration of 300 rpm/s. The specific rotation speed to be used depends on the thickness of the photoresist layer desired. This thickness will eventually be the depth of the channels of the microfluidic device to

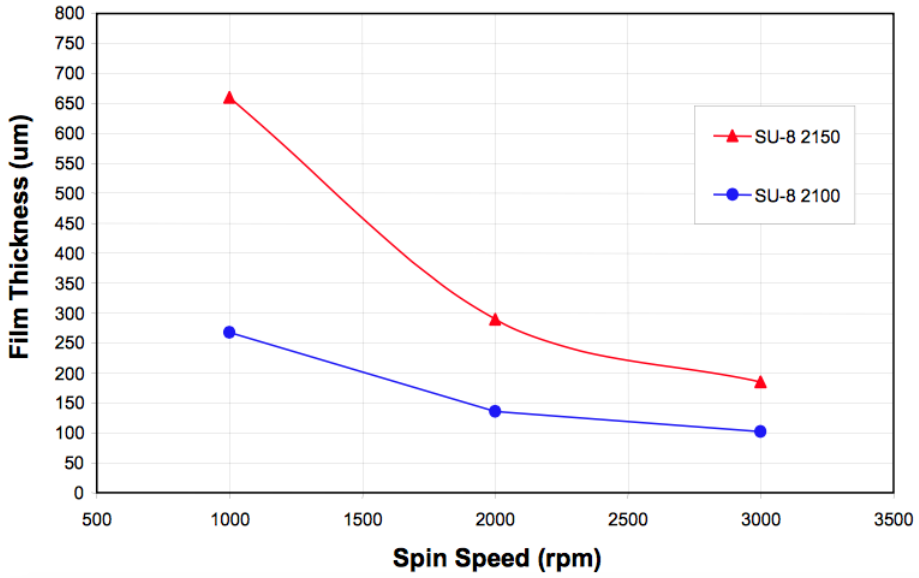


Figure 4.4: Relationship between final wafer rotation speed and the thickness of the SU-8 2100 photoresist layer desired. This figure is reproduced from Figure 1 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem. The blue line corresponds to the photoresist used in this project.

be created at a later stage. The relationship between the layer thickness and the rotation speed is illustrated in Figure 4.4. For the channels and respective wafer casts created for experiments in this chapter, 3000 rpm was chosen to get a photoresist layer thickness of 100 µm.

Bubbles formed in the photoresist layer during the spin process, particularly towards the centre of the wafer, were carefully removed using tweezers. The presence of bubbles in the photoresist layer would interfere with the baking steps later on.

Each coated wafer, with the photo-resist layer on top, was transferred to a level hot-plate, to allow a "soft bake" to take place. The purpose of this is to remove the solvent from the photoresist layer, allowing it to partially solidify. The bake also takes place in two stages; the first at 65 °C and the second at 95 °C. The time durations of these two stages are dependent upon the photoresist layer thickness — this is detailed in Table 4.1. The two-stage process ensures that the solvent does not evaporate too quickly from the upper surfaces and avoids the creation of a skin that would otherwise inhibit solvent evaporation from deep within the photoresist layer.

After the soft bake process, each wafer was taken off the hot-plate and allowed to cool to room temperature, before being placed on the hot-plate again for a few minutes. If wrinkles were observed in the photoresist layer of a wafer, the cycle of cooling and re-

Table 4.1: Relationship between duration of soft bake stages and the thickness of the SU-8 2100 photoresist layer desired. This table is reproduced with data from Table 2 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem.

Thickness / μm	Soft bake time / minutes	
	Stage 1 — 65 °C	Stage 2 — 95 °C
100 – 150	5	20 – 30
160 – 225	5 – 7	30 – 45
230 – 270	7	45 – 60
280 – 550	7 – 10	60 – 120

heating was performed on that wafer until no further wrinkles formed. A flat surface for the photoresist layer is necessary to ensure that any microfluidic channels formed using this cast have flat surfaces.

X-ray films with negatives of the channel designs were printed and cut to fit the wafer surface size. The photoresist layer of each wafer was exposed to UV light that had passed through the X-ray film with the negative of desired channel design. The exposure time required depends on both the photoresist layer thickness and the intensity of UV light used for the exposure. The energy doses required for various photoresist layer thicknesses are given in Table 4.2.

Table 4.2: Relationship between UV light exposure energy doses and the thickness of the SU-8 2100 photoresist layer desired. This table is reproduced with data from Table 3 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem.

Thickness / μm	Exposure energy / mJ cm^{-2}
100 – 150	240 – 260
160 – 225	260 – 350
230 – 270	350 – 370
280 – 550	370 – 600

In this work, a Karl Suss MJB3 Mask Aligner (Figure 4.5) was used with a UV-lamp with an intensity of $10\,mW\,cm^{-2} = 10\,mJ\,s^{-1}\,cm^2$. For a 100 μm layer, an exposure energy of $240\,mJ\,cm^{-2}$ was required. Therefore:

$$\text{Required exposure time} = \frac{\text{Exposure energy}}{\text{Intensity}} = \frac{240\,mJ\,cm^{-2}}{10\,mJ\,s^{-1}\,cm^2} = 24\,s.$$

UV light causes the chemicals in the photoresist to polymerise. Thus the areas of the photoresist layer that polymerise should be in the shape of the channel design printed in the X-ray film.

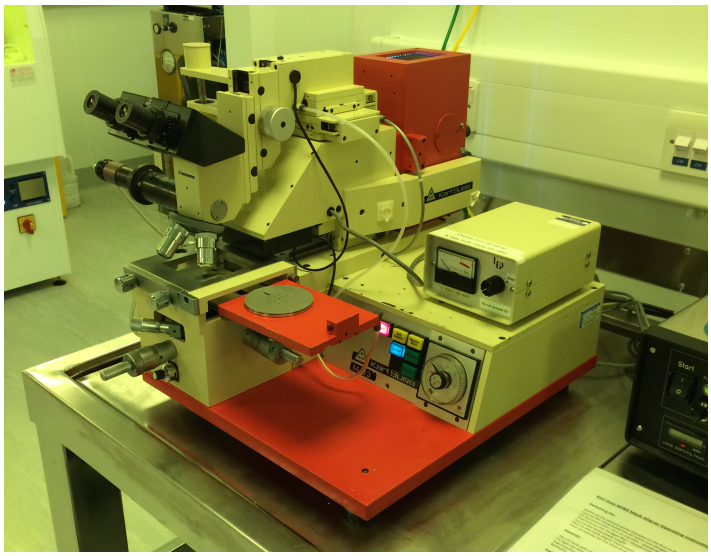


Figure 4.5: A photo of a Karl Suss MJB3 Mask Aligner, used for the exposure of silcon wafers with a photoresist layer to UV light. The mask is placed, photoresist layer on top, on the silver circular holder on the red plate. This is then moved into place under a glass in the exposure zone (located underneath the microscope). A vacuum holds the silicon wafer in place. The X-ray film with the design printed as a negative is placed on the glass just underneath the microscope — the "shiny" film side should be placed facing up.

Immediately after exposure to UV light, each wafer, with the photo-resist layer on top, was transferred to a level hot-plate, to allow a post exposure bake to take place. Similar to the soft bake, this also takes place in two stages; the first at 65 °C and the second at 95 °C. The time durations of these two stages are dependent upon the photoresist layer thickness — this is detailed in Table 4.3.

Table 4.3: Relationship between duration of post exposure bake stages and the thickness of the SU-8 2100 photoresist layer desired. This table is reproduced with data from Table 5 of the Processing Guidelines for SU-8 2100 and SU-8 2150 photoresists by Microchem.

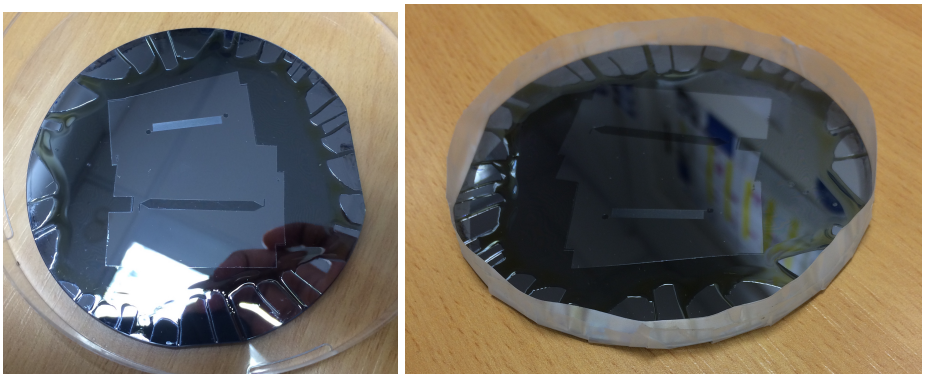
Thickness / μm	Post exposure bake time / minutes	
	Stage 1 — 65 °C	Stage 2 — 95 °C
100 – 150	5	10 – 12
160 – 225	5	12 – 15
230 – 270	5	15 – 20
280 – 550	5	20 – 30

After this baking stage, each wafer was left to cool down, before being washed in SU-8 developer fluid with agitation using compressed air. The developer fluid washes away any parts of the photoresist that are unpolymerised (*i.e.* unexposed to UV light). This step leaves a section of photoresist on the wafer in the shape of the microfluidic channel design. The time taken to "develop" the photoresist increases with increasing layer thickness, from 15 minutes for 100 μm thick layers to 40 minutes for 550 μm thick layers.

After all regions of unpolymerised SU-8 were removed, each wafer was cleaned with an isopropanol rinse and dried with compressed air.

4.3.2 Manufacture of the PDMS microfluidic channel from the cast

To form a microfluidic channel from PDMS using a wafer cast with the required channel design, Scotch® tape was applied to the silicon wafer edge and underside to create a well. This well allows PDMS to be poured into the well and set over the channel design.



(a) Wafer cast.

(b) Wafer cast with Scotch® tape on the edge to form a well.

Figure 4.6: An example of a finished silicon wafer cast with channel designs in photoresist in the centre. Scotch® tape is attached to the wafer to form a well for the liquid PDMS to set.

The wafer was placed on a hot-plate, with heating turned off, and with the wafer flat and level, to enable the PDMS to set with even thickness.

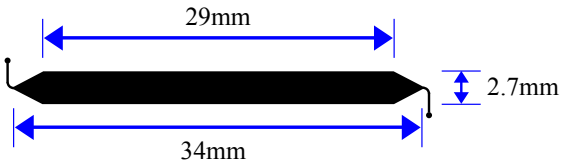
A mixture of liquid PDMS and curing agent was prepared with a 10:1 mass ratio and stirred thoroughly and air bubbles were removed using a vacuum dessicator. The mixture was carefully poured into the wafer well on the hot plate, with all parts of the plate covered evenly and without bubbles being formed. The hot-plate was set to 85° to 90° and the PDMS was left to cure and set.

After this process, the PDMS could then be cut and peeled slowly from the wafer. The section of PDMS with the channel design embedded was cut out with a scalpel, and holes bored, using a biopsy puncher, into the channel entry points, to allow fluid to enter and exit the channel.

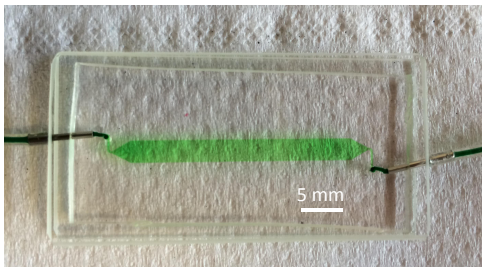
4.4 Preliminary experiments with microfluidic devices

Two initial designs of microfludic channel were tested with DNA solutions to see if an LD signal could be obtained. The designs are:

- *"Wide channel"* — This design, illustrated in Figure 4.7 is intended to test a pressure-driven flow where the whole beam from a CD spectropolarimeter passes through the solution as it flows through.
- *"Zig-zag channel"* — This design, illustrated in Figure 4.8, is intended to test a pressure-driven flow where the fluid is contained in a very narrow channel and passes multiple times through the path of the beam of a CD spectropolarimeter.

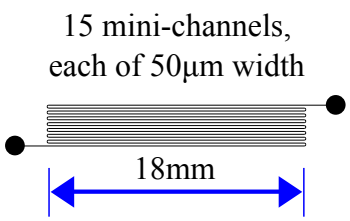


(a) Design drawing

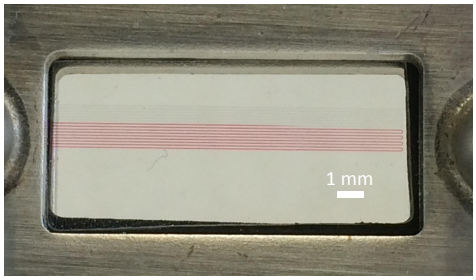


(b) Channel filled with green fluid

Figure 4.7: "Wide Channel"



(a) Design drawing

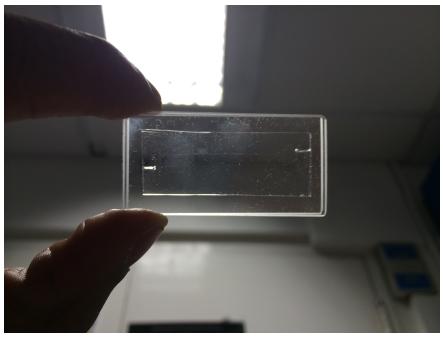


(b) Channel partially filled with red fluid

Figure 4.8: "Zig-zag channel"

The channels were manufactured in 2 mm thick PDMS and placed between two 1 mm thick quartz microscope slides (see Figure 4.9a). Needle tips from 0.5 inch guage needles were cut and placed in sections of Tygon® microbore tubing, 0.020 inches inner diameter and 0.060 inches outer diameter. These were then inserted into the sides of the PDMS, as shown in Figure 4.7b). This assembly was then placed in a holder for an IR spectroscopy cell, to keep the assembly together and prevent leaking of sample fluid from the channel.

A solution of 1 mg ml^{-1} calf-thymus DNA in 10 mM pH7.4 Sodium Phosphate buffer (prepared using protocols in [125]) was pumped through the two devices using a syringe pump.



(a) Channel placed between two quartz microscope slides.

(b) An IR spectroscopy cell holder

(c) Inside an IR cell holder.

Figure 4.9: Illustration of the use of an IR spectroscopy cell holder for the flow cell.

4.4.1 Results with wide channel

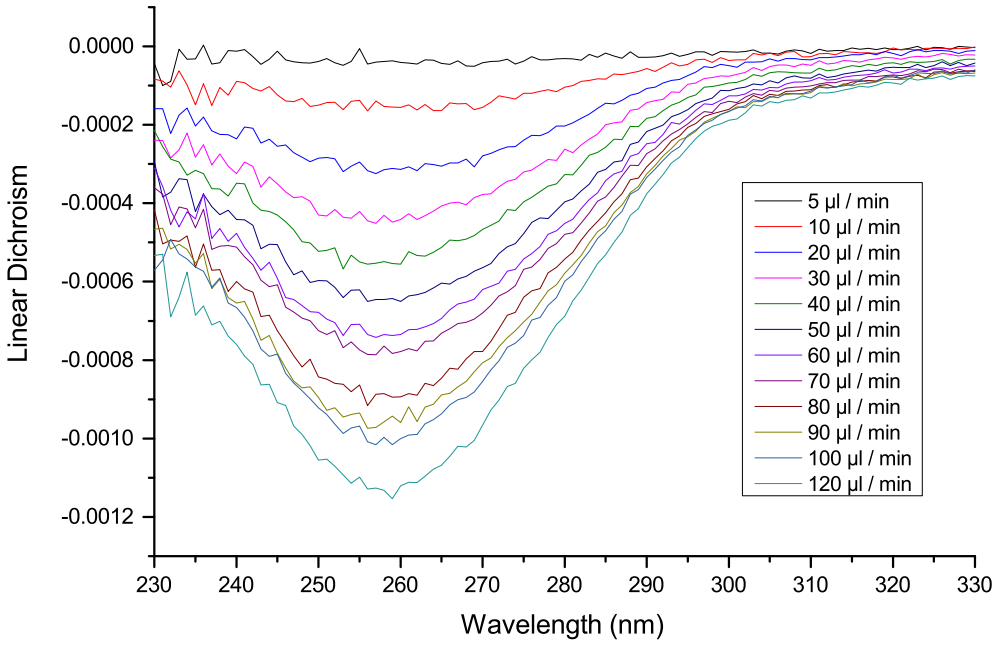


Figure 4.10: Linear dichroism spectra of DNA solution passed through the wide channel microfluidic device at different flow rates.

The main features of the linear dichroism spectra (Figure 4.10) are the negative peaks at around 260 nm, which correspond to transition moments along the planes of bases within DNA molecules. As these are aligned perpendicular to the long axis of DNA molecules and, therefore, also perpendicular to the flow direction, these electronic transitions give rise to the negative LD peaks in Figure 4.10.

Furthermore, the peaks at 260 nm in the LD spectra become progressively more negative as the flow rate increases. Upon dividing the linear dichroism by the absorbance (Figure 4.11), we obtain the reduced linear dichroism (Reduced LD). These spectra are shown in Figure 4.12 and indicate this trend of stronger LD signal with increased flow rate, suggesting better alignment when flow rates are higher. However, this trend is less discernable at wavelengths of 290 nm and higher, as the signal-to-noise ratio in this region of the LD spectra increases.

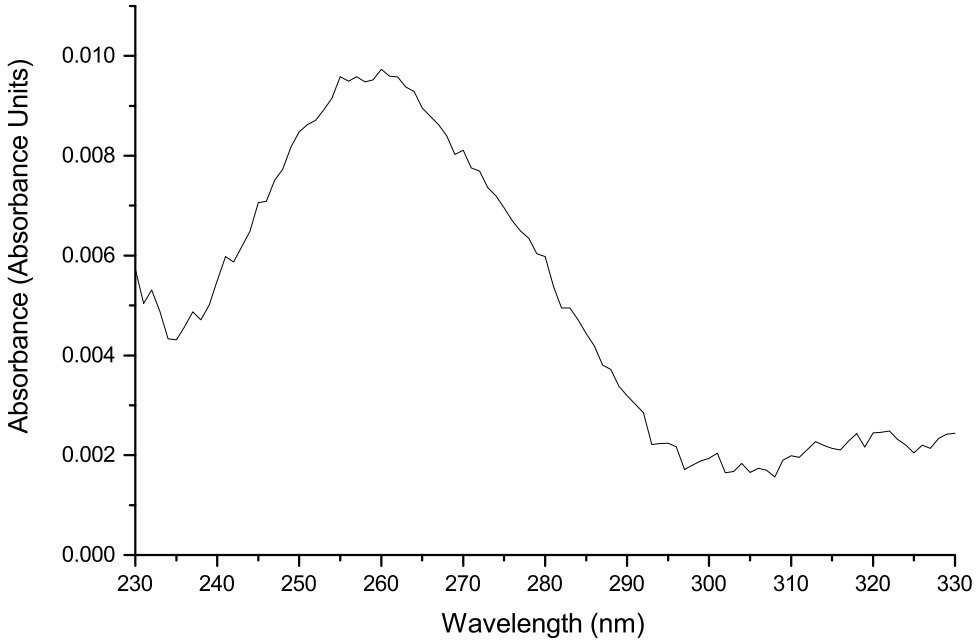


Figure 4.11: Absorbance of DNA solution passed through the wide channel microfluidic device.

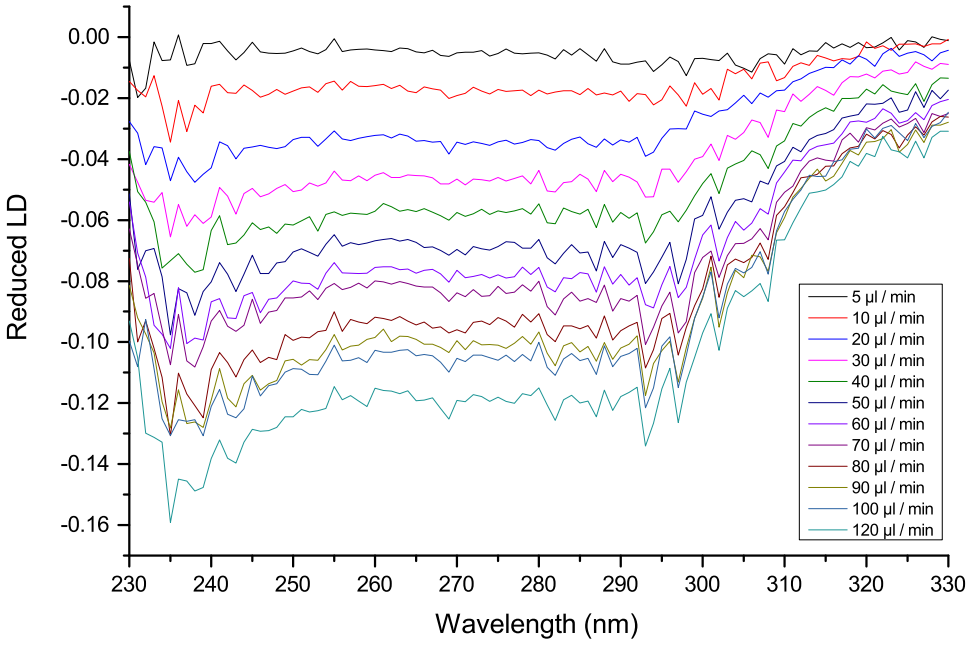


Figure 4.12: Reduced linear dichroism spectra of DNA solution passed through the wide channel microfluidic device at different flow rates.

To study how the theoretical alignment changes with increased flow rate, recall that the LD signal is proportional to a function of the angle between the electronic transition and the

alignment axis, the isotropic absorbance of the sample and the proportion of molecules that are aligned in the sample. Reduced LD removes the dependence on isotropic absorbance. Furthemore, the orientation parameter value can be estimated from reduced LD data by assuming that the effective angle between the flow axis and the chromophore responsible for absorbance at 260 nm is 86° [96, 97] and applying Equation (1.1.5). Therefore, plotting the reduced LD against the flow rate will give a rough indication of the performance of the microfluidic device, since a higher flow rate leads to stronger wall shear forces.

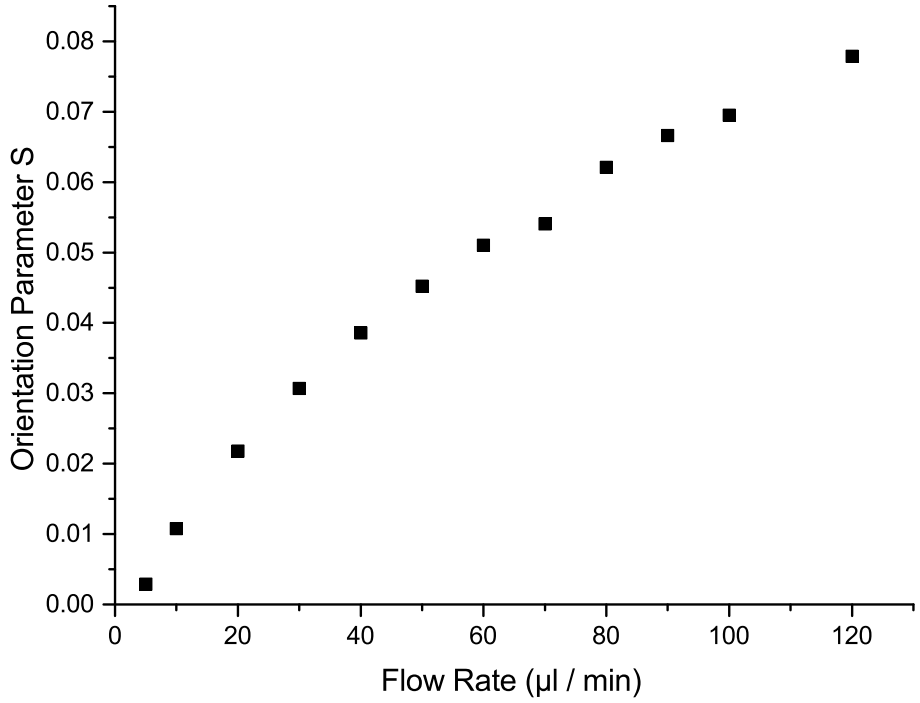


Figure 4.13: Estimated values of the orientation parameter (calculated using the reduced LD signal measured at 260 nm) against flow rate using the wide channel microfluidic device.

Figure 4.13 indicates that increases in the flow rate lead to increased alignment of DNA molecules in the sample. However, at higher flow rates, increases in the alignment are not as strong with increasing flow rate.

4.4.2 Results with zig-zag channel

The absorbance, linear dichcroism and reduced LD spectra obtained using the "zig-zag" channel are shown in Figures 4.14 to 4.16 respectively. While the absorbance measured is of a similar magnitude to that obtained with the wide channel device, the linear dichroism spectra are noisier and the readings are two orders of magnitude smaller in scale. The reduced LD spectra are, therefore, very noisy and small in scale.

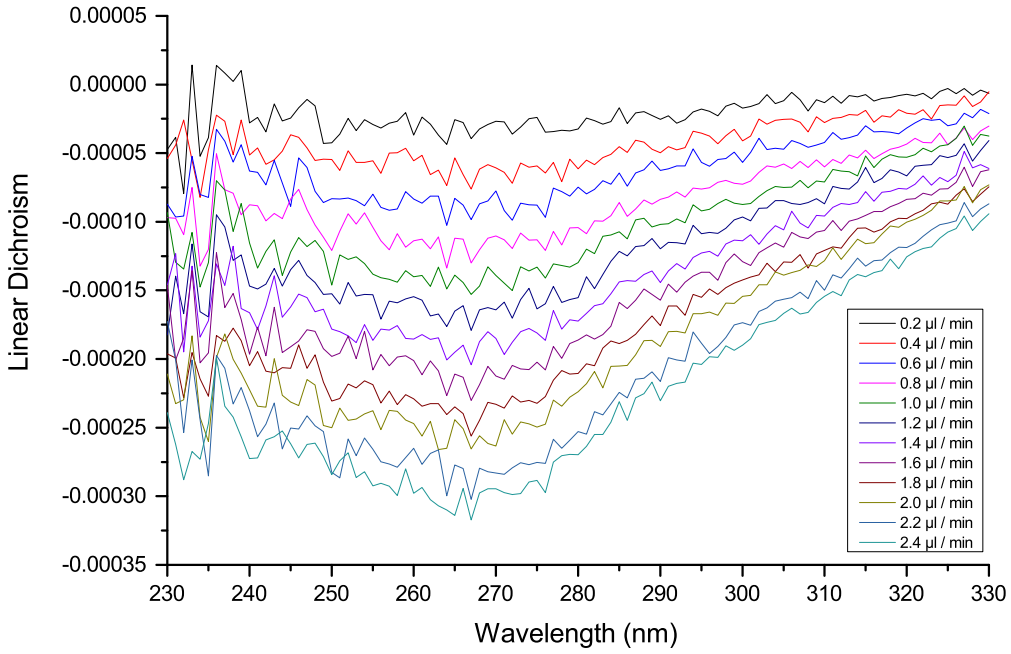


Figure 4.14: Linear dichroism spectra of DNA solution passed through the zig-zag channel microfluidic device at different flow rates.

Following a similar approach to the wide channel data, plotting the estimated orientation parameter (based on the reduced LD signal at 260 nm) against the flow rate, as shown in Figure 4.17, gives a similar trend to that of the wide channel, where increased flow rates give rise to an increase in the orientation parameter.

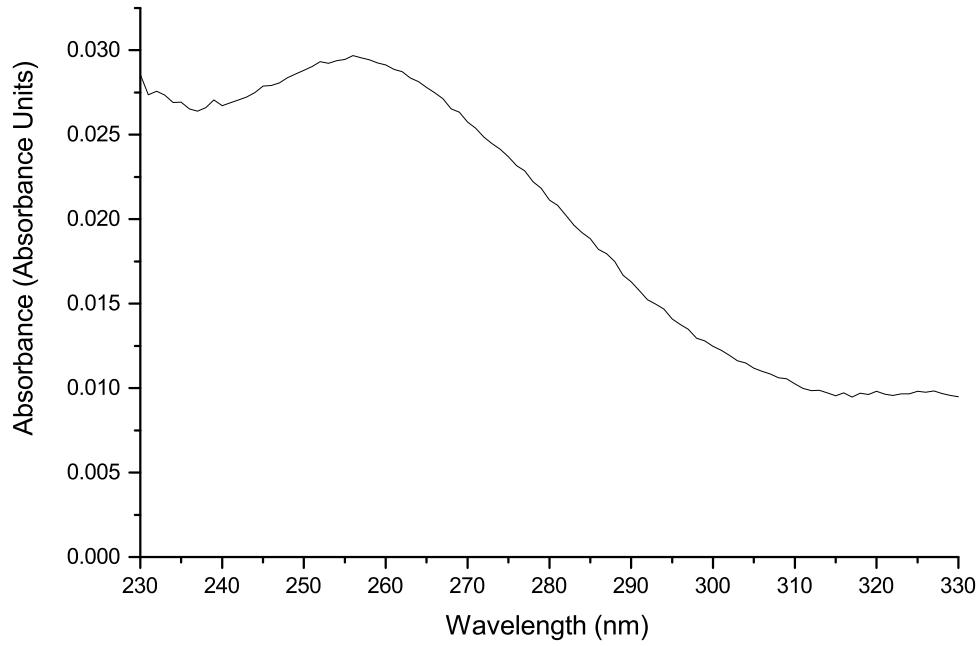


Figure 4.15: Absorbance of DNA solution passed through the zig-zag channel microfluidic device.

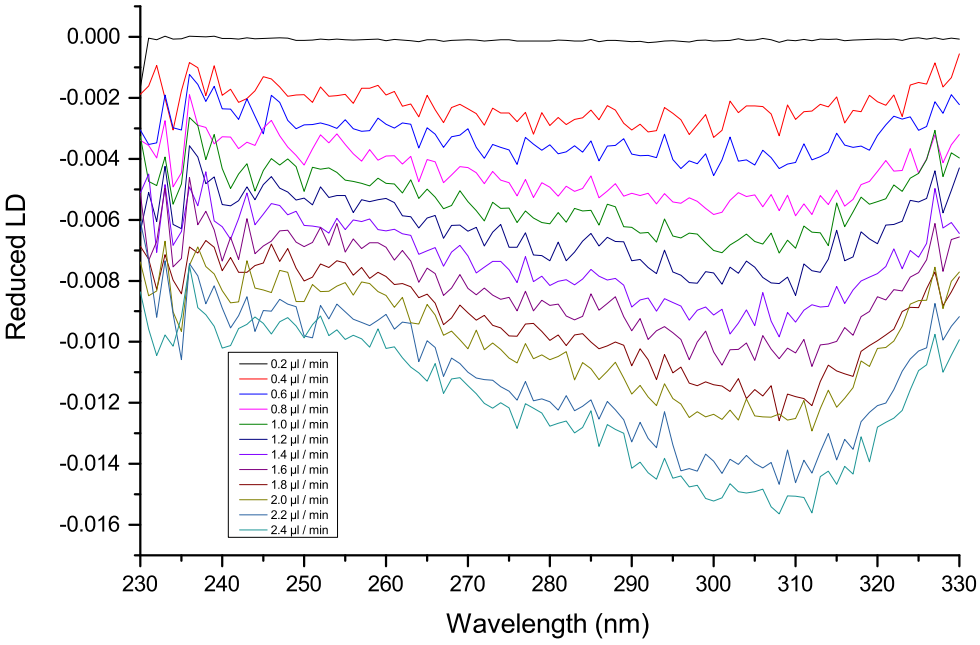


Figure 4.16: Reduced linear dichroism spectra of DNA solution passed through the zig-zag channel microfluidic device at different flow rates.

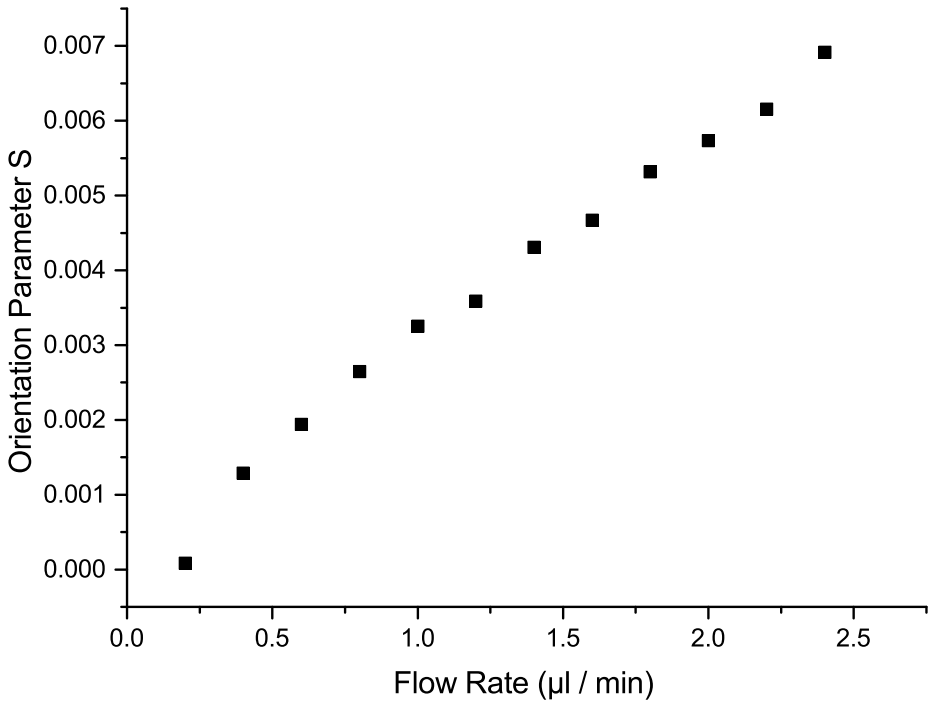


Figure 4.17: Estimated values of the orientation parameter (calculated using the reduced LD signal measured at 260 nm) against flow rate using the zig-zag channel microfluidic device.

4.4.3 Analysis of results

The general trend in both devices is of increasing strength of linear dichroism signal with increased flow rate. This indicates that, as expected, alignment increases with flow rate. However, the LD spectra measured in both devices and the absorbance spectrum for the wide channel have a low signal-to-noise ratio, suggesting that the devices could benefit from being constructed with deeper channels. This would increase the path lengths and, therefore, the magnitude of the absorbance and LD signals to overcome noise present in the measurements. The high signal-to-noise ratio observed in the LD and reduced LD spectra obtained with the zig-zag channel demonstrates the difficulty in attempting to obtain a clear LD spectrum from a solution with molecules that can be oriented in flow. These results, combined with the orientation parameter values of an order of magnitude less than that obtained with the wide channel, meant that the zig-zag channel design was not taken forward.

4.5 Development of an extensional flow device

In seeking to try to align DNA molecules and vesicles (to be discussed in the next chapter) in flow, a planar extensional flow setup was considered. In this type of flow, fluid is plushed into a central chamber along one axis and is removed along a perpendicular axis. The simplest setup that gives rise to extensional flow is a cross-slot (see Figure 4.18). Polymer molecules present in the central chamber are stretched along the fluid exit axis; it is this stretching that we attempt to use to align molecules for LD experiments.

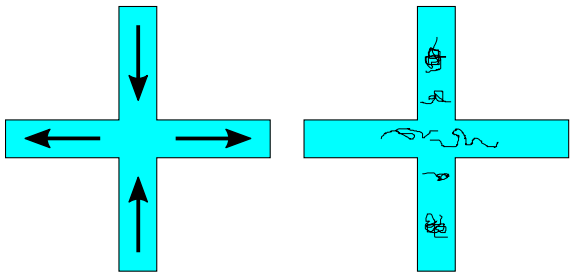


Figure 4.18: Basic schematic of a cross-slot with extensional flow in the centre. The left diagram shows the flow directions. The right diagram illustrates the behaviour of polymer molecules in extensional flow.

Previous experiments have been carried out by others to investigate the behaviour of DNA molecules in extensional flows [6, 127, 128, 159], but none so far have attempted to obtain absorbance spectroscopic data from samples aligned in this way. The aim for us was to develop a microfluidic device where planar extensional flow can be utilised, with a chamber small enough to generate significant extensional strain on molecules to align them as desired, but large enough to allow a beam from a CD spectropolarimeter to pass through the fluid sample where the molecules are aligned.

An issue with a simple cross-slot, as depicted in Figure 4.18 is that the extension rate is well-defined only at the stagnation point in the centre of the cross; furthermore, the extensional strain is concentrated tightly around this point [43, 44]. To expand the region of fluid exposed to extensional strain, Alves [4] proposed a numerical method to design an optimized cross-slot that would result in a constant, homogeneous extension rate along the fluid entry and exit axes.

This design, shown in Figure 4.19, was used in Haward et al. [44] to investigate the behaviour of poly(ethylene oxide) molecules in extensional flow. While further investiga-

tions into more detailed designs optimised for extensional flow studies in three-dimensions were devised during the course of this investigation [39], it was decided that this design should form the basis of our extensional flow channel. Further details of the design are given in Figure 4.20.

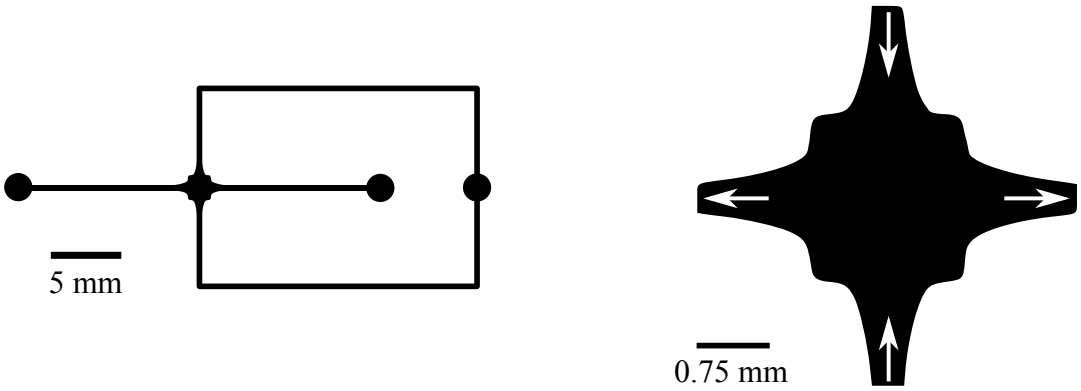


Figure 4.19: The design for the extensional-flow microfluidic device used in experiments in this thesis. The left diagram is the complete design of the channels embedded in PDMS. The right diagram is a zoomed-in view of the central intersection, with arrows indicating the flow directions.

The original design of the channel in [44] is two thirds size of the design used here. This is because it was decided to increase the size of the central intersection to allow as much of the CD spectropolarimeter beam to pass through it. Furthermore, it was difficult to position the chamber in such a way the beam would pass through the device in exactly the same spot; increasing the size of the intersection would allow for slight changes in the device positioning. A lens was also used with a focal length of 6 cm. The beam produced is collimated and has a rectangular profile and so can be focused onto a thin strip within the cross-slot, along the fluid exit axis.

The nature of the channel design in Figure 4.19 means that use of an IR cell holder is not suitable. Tubes must be inserted into the device perpendicular to it rather than from the side as for the preliminary tests. To enable this revised setup, a method was found of securing the PDMS channels to quartz microscope slides with oxygen plasma.

- The PDMS channel and the quartz slide which are to be bonded together are cleaned with Scotch® tape and compressed air respectively.
- Both are placed, with bonding surfaces facing up, inside a plasma etcher. They are

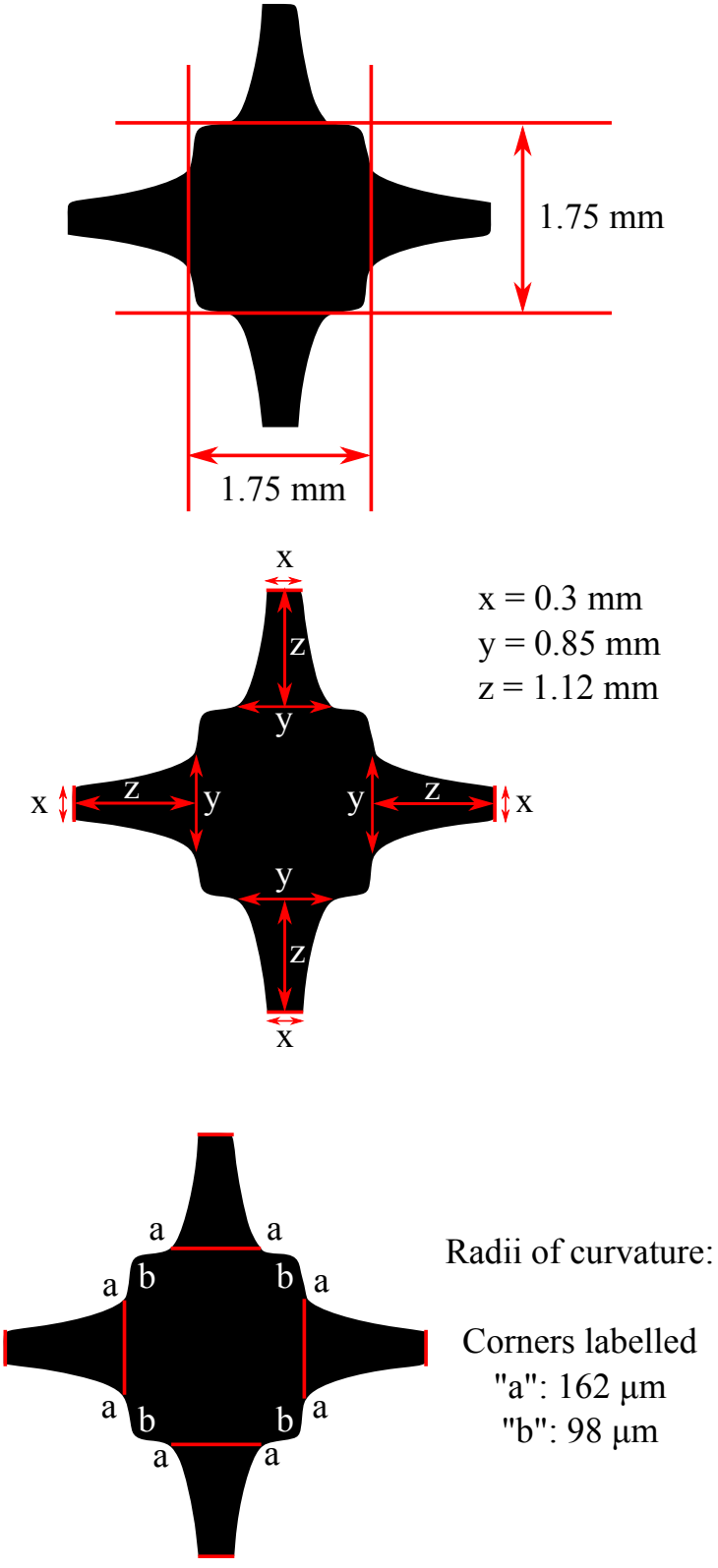


Figure 4.20: Details of particular lengths and radii of curvature for the Haward cross-slot design used in the experiments reported in this thesis.

treated with oxygen plasma for 10 minutes to clean the surfaces and to decrease the hydrophobicity of the PDMS. Both of these enable the two to bind more securely [145].

- The channel and microscope slide are bonded together and the ensemble is secured by leaving it on a hot plate at 80 °C for 3 hours.

Biopsy punchers are used to bore holes perpendicular to the plane of the PDMS. The tubes carrying the fluid sample are inserted into these holes and secured into place by placing a cut segment of a micropipette tip around the entry point of the tube and applying a small amount of liquid PDMS (already premixed with curing agent) around the joint. This mitigates against leaks forming at the point where fluid enters or exits the device.

Finally, the syringe pump used in earlier preliminary experiments was replaced with a high-performance liquid chromatography (HPLC) pump. This enabled the recirculation of sample fluid through the microfluidic device, avoiding the preparation of large sample volumes for each experiment.

4.6 Measuring and modifying the path length of microfluidic devices

The absorbance and, therefore, linear dichroism measured from a sample are proportional to the path length — the thickness of sample which the light beam passes through. One problem indicated by the preliminary experiments is the low signal-to-noise ratio. A possible explanation for this is a low path length for the beam to pass through the sample, leading to low absorbance and LD signals.

Furthermore, our experiments used Couette flow as a positive control to establish alignment of DNA in solutions is possible. These use an effective patch length of 500 μm. It is arguably more sensible to consider channel depths that are of the same order of magnitude as the fluid dynamics involved are more comparable against each another. This is because the scale of a fluid flow is known to change the prominence of certain forces — as the length scale reduces to the microscale, surface force such as shear forces become more prominent than volume forces [17, 144].

In addtion, while the speed at which the wafer with photoresist is spun theoretically determines the thickness of the photoresist layer (and therefore the channel depth), discrepancies can occur. For example, the wafer might not be centred correctly before photoresist liquid is placed on it, leading to an uneven photoresist layer forming when the wafer is spun. Here, we briefly outline how channel depths were increased in both the wide channel and cross-slot devices. We also outline two methods used to check the depth of the new microfluidic devices; Non-contact profilometry, using light interferometry, and absorbance measurements, using potassium chromate.

4.6.1 Using light inteferometry to measure channel depth of devices

Light inteferometry uses the superposition of light waves originating from a light source with a known location. The surface profiler (Bruker Contour GT) used in these experiments uses a height adjustable light source. The initial position of this source is such that a light interference pattern can be seen on upper surfaces of the objects being profiled. During a scan, the light source is lowered and the machine detects when an interference

pattern is present on a lower surface. The machine uses the distance travelled by the light source between the appearance of the two interference patterns to determine the depth of structures in the object being scanned.

The area to be measured is divided up into overlapping zones by the software; each zone covers an area over the machine can scan without moving the light source horizontally. All the zones are then stitched together to form a composite depth measurement map.

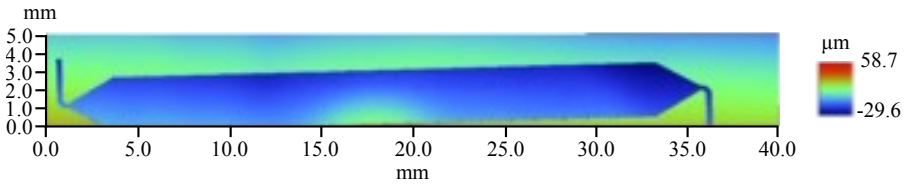


Figure 4.21: Surface profiler depth map of the orignal wide channel used for preliminary tests.

Figure 4.21 displays the surface profiler measurements of the wide channel used for the preliminary LD experiments with DNA. Examination of the data indicate that the depth of the channel varies between 30 µm and 40 µm. The variation in measurements are down to a number of factors, including:

- The PDMS was not lying flat on the sample holder platform of the interferometer. This could be due to small twists and bends in the PDMS due to its flexibility or deformations in the channel during manufacture. In particular, in the curing phase of the channel manufacture, the upper surface of the PDMS may not have set exactly parallel to the bottom surface that was in contact with the silicon wafer.
- The tilt applied to the holding platform of the interferometer. In setting up the instrument to measure the channel depth, the platform is tilted slightly along two axes; this is done to maximise the recognition of the light interference waveforms that appear on the surfaces of the PDMS during the measurement.
- During the manufacture of the silcon wafer mould, the photoresist may not have set evenly. One known problem is the issue of wrinkling of the photoresist surface, if the soft bake and pre-exposure bakes have not taken place for long enough.

However, even with these issues, the measurements indicated that the devices were an order of magnitude smaller in path length, than that of the microvolume Couette flow

cells used in our lab. This was thought to explain why the signal-to-noise ratio of the data obtained with the wide channel device was smaller than that obtained with Couette flow.

4.6.2 Increasing path length and measuring channel depths using light interferometry

The protocol for manufacturing the channel cast was modified by repeating certain steps in the application of photoresist to the silicon wafer.

- After one layer of photoresist was applied to the wafer and spun to the desired thickness and soft-baked, another layer of photoresist was applied on top of this layer in exactly the same way. Each layer was spun to give each a thickness of roughly 250 μm, giving a total thickness of 500 μm.
- The exposure of the wafer to UV light takes place as in the protocol; however, the exposure time is extended to allow for the increased thickness of the photoresist layer. As we have aimed for a thickness of 500 μm, extrapolation of the protocol instructions gives an exposure time of 1 minute.

Furthermore, at this point a redesign of the wide channel, shown in Figure 4.22, was carried out to shorten the channel. The choice of length was based on the dimensions of a similar cell used by Matsuo et al. [84] for LD studies of membrane proteins. The channel was also widended to ensure that all of the light from the CD spectropolarimeter would pass through the fluid sample in the channel. The design of the cross-slot remained unaffected.

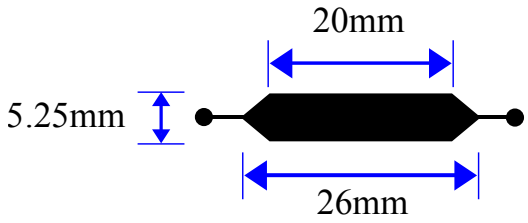


Figure 4.22: Dimensions of the revised wide channel design.

After generating the thicker cast and forming the channels from PDMS, measurements

were taken with the interferometer. These are displayed in Figure 4.23 for the wide channel and Figure 4.24 for the cross-slot.

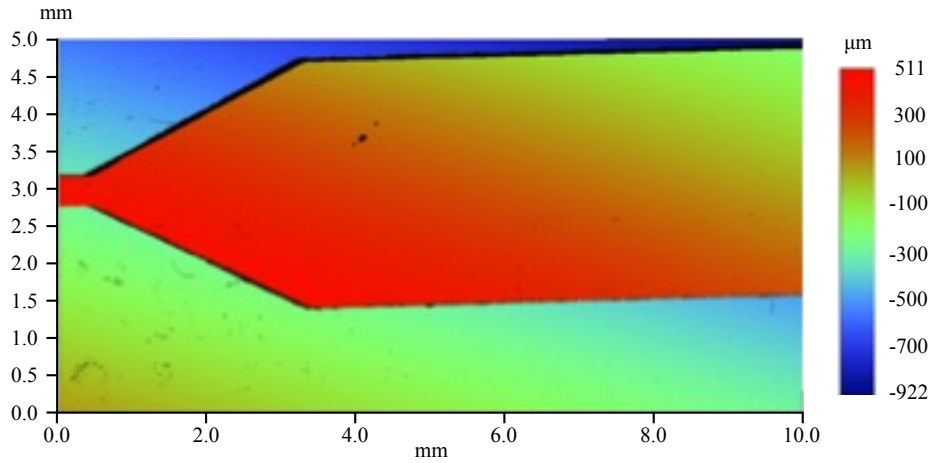


Figure 4.23: Surface profiler depth map of the deeper wide channel design on the silicon wafer, as used for later experiments. The tilt that can be observed here is due to the configuration of the platform on which the wafer was placed, so as to allow the inteferometer to capture the light interference waveforms precisely.

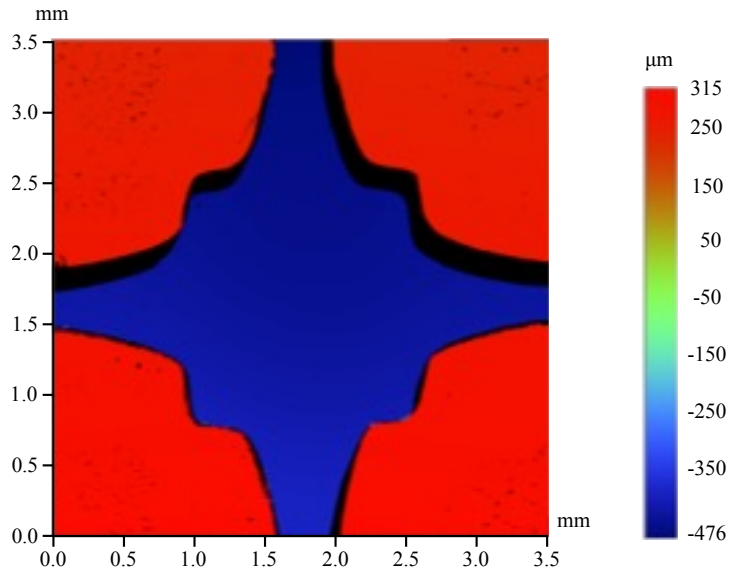


Figure 4.24: Surface profiler depth map of the deeper cross-slot used for final tests.

The surface profile measurements indicated that the depth of the deeper wide channel was roughly 740 µm and the deep cross-slot was approximately 730 µm in depth. This is more than the 500 µm targetted and is likely to be due to the difficulties in controlling the thickness of the photoresist applied to the silicon wafer (particularly when applied as two separate layers).

4.6.3 Using potassium chromate absorbance to measure channel depth

Physical measurements, such as those obtained by profilometry, provide a static assessment of the channel depth. However they do not provide an indication of the behaviour of a channel *in situ* when absorbance and LD measurements are being taken. For example, while fluid is pumped through a device, the PDMS material may deform due to the fluid pressure, resulting in an increase in the channel depth and, therefore, the path length of fluid sample. Thus an increase in the LD signal with increasing flow rate might be more to do with the increase in absorbance due to path length changes.

Use of potassium chromate solutions is common for calibration in absorbance spectroscopy. In particular, when solutions are prepared to a desired concentration with precision, they can be used to make accurate determinations of the path length of cuvettes to be used for absorbance spectroscopy. This, therefore, is a suitable setup for assessing any dynamic channel depth changes in microfluidic devices, such as those outlined above.

Cross-Slot Device

A 2 mM potassium chromate solution was pumped through the device using an HPLC pump. The spectrum shown in Figure 4.25 was taken with the flow rate at zero.

The extinction coefficient of potassium chromate at 372 nm, ε_{372}, is 4830 $\mathrm{mol^{-1}\,dm^{-3}\,cm^{-1}}$. Therefore, using the Beer-Lambert law (Equation (1.1.2)),

$$\text{Path length} = \frac{A_{372}}{C\varepsilon_{372}} = \frac{0.521137}{2 \times 10^{-3} \times 4830}\text{cm} = 540\,\mu\text{m}$$

Wide Channel

During experiments which used the deeper wide channel device, it was observed that the "ceiling" of the channel was bulging out as the flow rate was increased, due to the flexibility of PDMS. To see how this affected the apparent path length of the device, absorbance spectra of 2 mM potassium chromate solution passing through the device were measured with different flow rates in use. These can be seen in Figure 4.26.

Using the path length calculation used for the cross-slot device, the derived path lengths

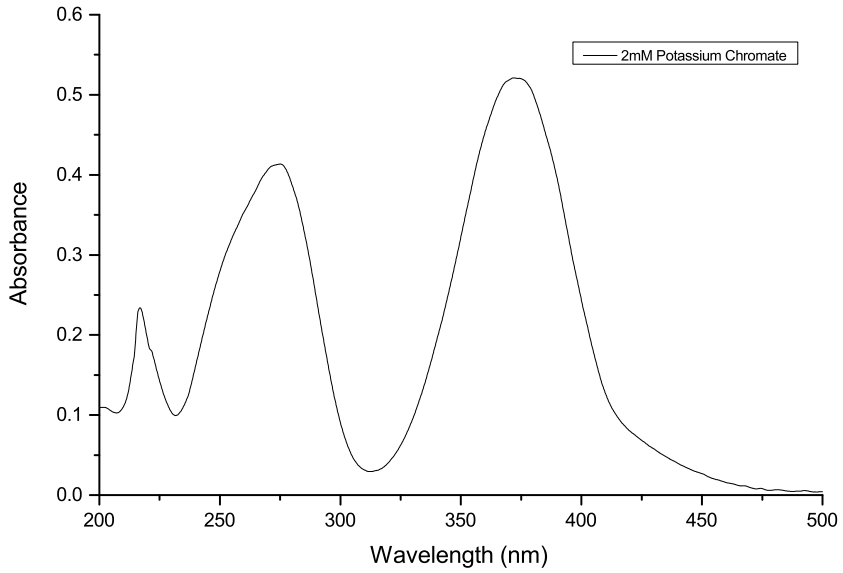


Figure 4.25: Absorbance spectrum of 2 mM potassium chromate solution passes through the cross-slot device.

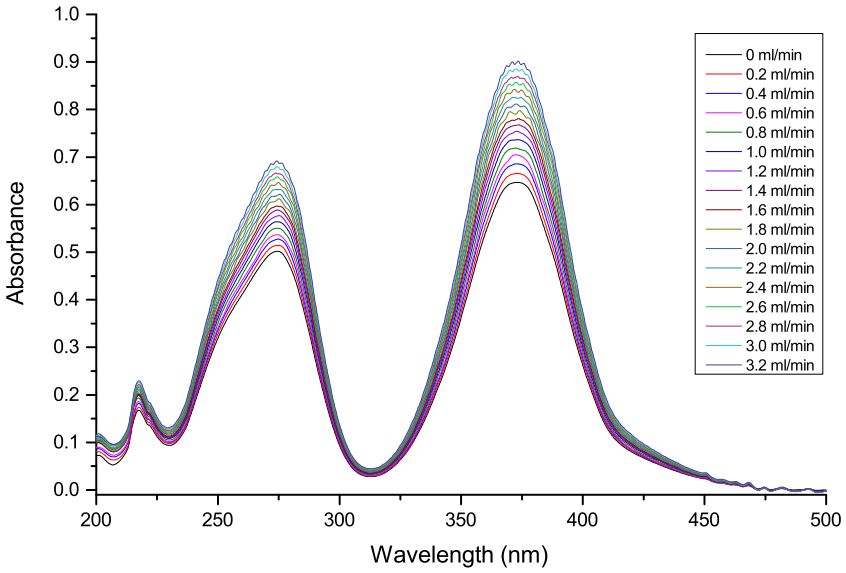


Figure 4.26: Absorbance spectra of 2 mM potassium chromate solution through the wide channel microfluidic device as it is pumped at different flow rates.

of the wide channel are shown in Figure 4.27. Since the channel did not maintain a rectangular cross-section, the values given in Figure 4.27 are not necessarily a true reflection of the path length of the device, as the depth will vary with position across the channel.

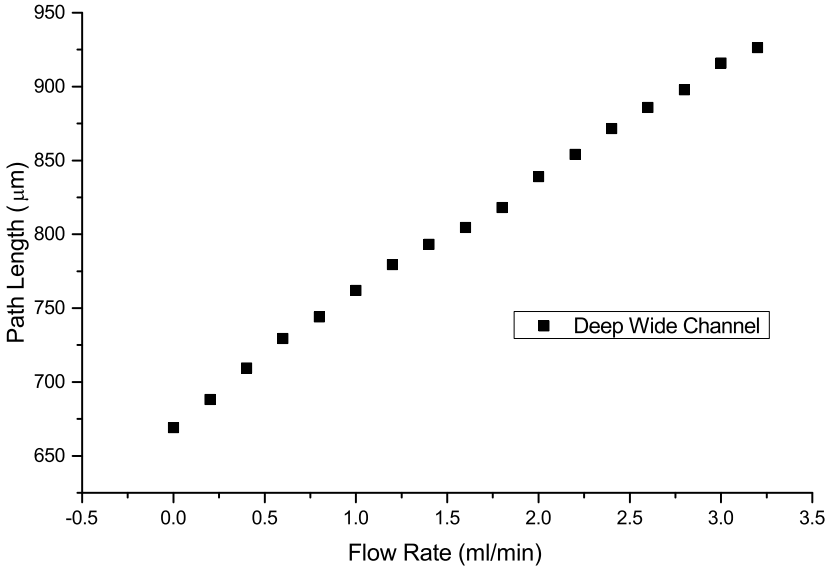


Figure 4.27: Derived path length of the wide channel microfluidic device at different flow rates.

4.7 Experimental results with deeper microfludic devices

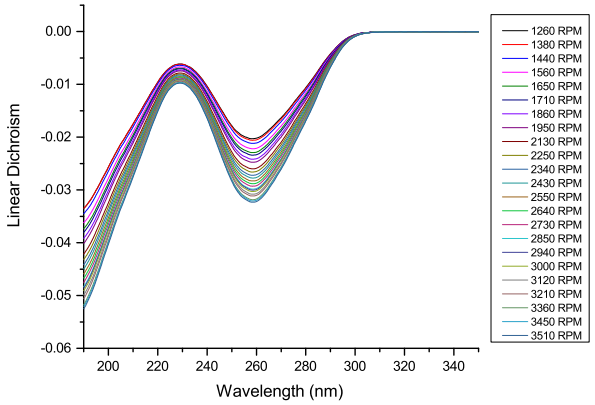
DNA solutions of 800 μM were made up by dissolving 2.6 mg of calf-thymus DNA (obtained from Sigma Aldrich) into 10 ml pH7.4 10 μM Sodium Phosphate buffer (prepared using protocols given in Russell and Sambrook [125]). Samples were checked for alignment in Couette flow before being pumped through the microfluidic device.

While processing the data, it was found that regular wave patterns were found in the LD spectra measured with both microfluidic devices and the absorbance spectra measured with the wide channel device. This was determined to be an artifact from the regular pumping action of the HPLC pump. Traces of this were removed using a low pass fast Fourier transform filter, removing waveforms with frequencies above 0.05 Hz, in OriginLab®.

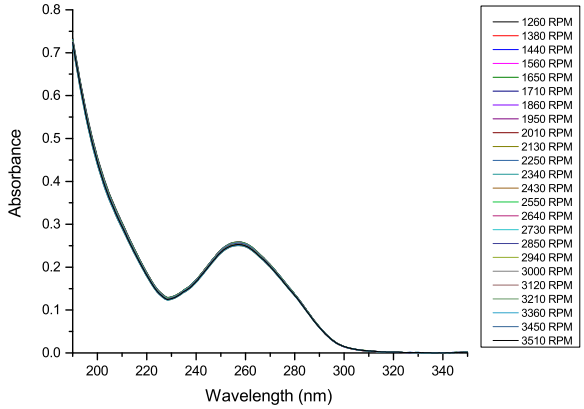
4.7.1 Couette shear flow

Figure 4.28 displays linear dichroism, absorbance and reduced LD spectra of a DNA solution in Couette shear flow at different rotation speeds. A stronger LD signal at 260 nm is seen with increasing rotation rate. The reduced LD signal reaches -0.13 at the highest

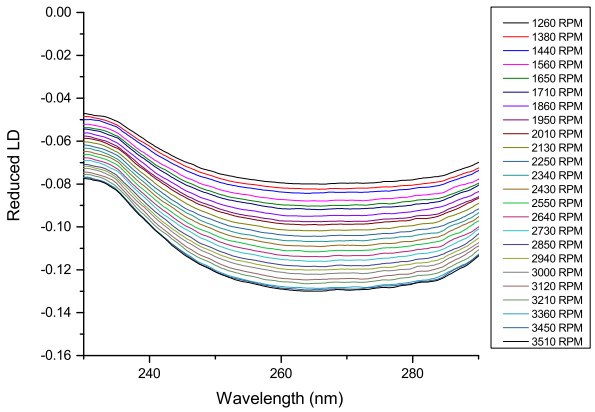
rotation rate of 3510 rpm.



(a) Linear dichroism



(b) Absorbance

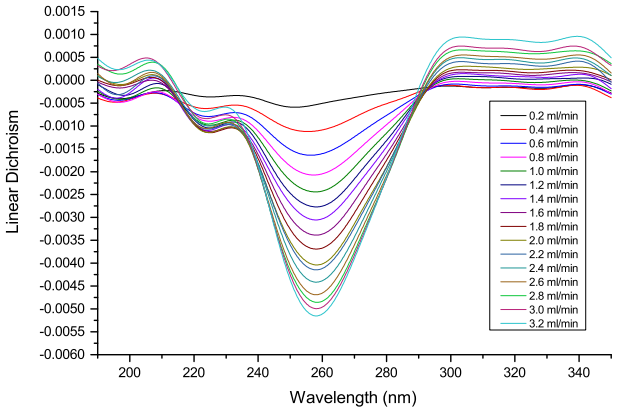


(c) Reduced LD

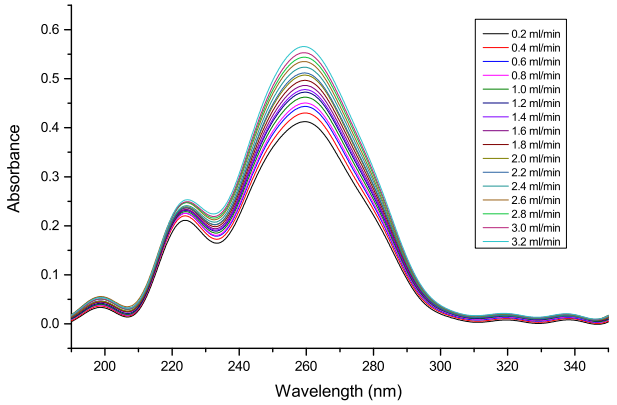
Figure 4.28: Linear dichroism, absorbance and reduced LD spectra of DNA solution in Couette flow at different rotation speeds. The measurements were taken with DNA solution prepared for use in the deep cross-slot microfluidic device. The results for the solution prepared for use in the deep wide channel device gave similar value for the reduced LD.

4.7.2 Poiseuille (pressure-driven) flow in the deep wide channel

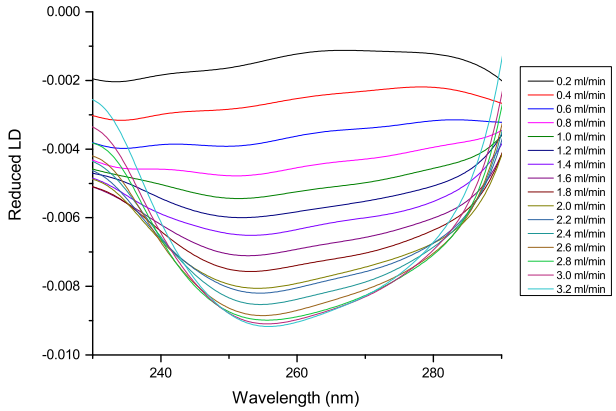
Figure 4.29 displays linear dichroism, absorbance and reduced LD spectra of DNA solution passed through the deep wide channel microfluidic device. The LD and reduced LD data obtained with this device are an order of magnitude lower than for Couette flow (Figures 4.28a and 4.28c) but still demonstrate the same pattern of increased fluid flow resulting in more molecule orientation and a more negative value for the LD and reduced LD. The signal-to-noise ratio is also much improved on the original wide channel data, due to a combination of the use of a higher concentration of DNA and the increased path length. As a result, absorbance values of 0.4 and above were measured with this device, compared with 0.01 measured using the original wide channel (Figure 4.11). However, the reduced LD signals obtained here are lower in magnitude than for the shallower channel used earlier.



(a) Linear dichroism



(b) Absorbance



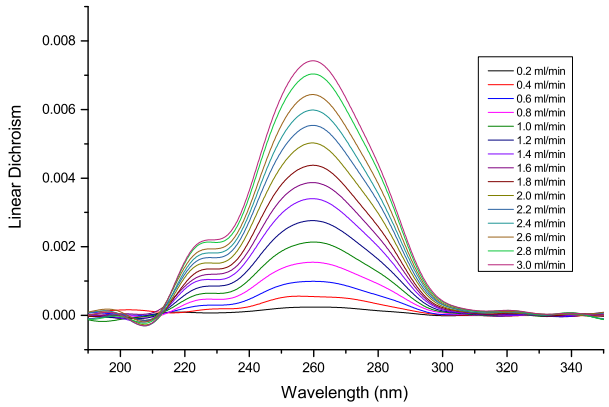
(c) Reduced LD

Figure 4.29: Linear dichroism & reduced LD spectra of DNA solution in pressure-driven flow through the deep wide channel microfluidic device and a plot of reduced LD at 260 nm against flow rate.

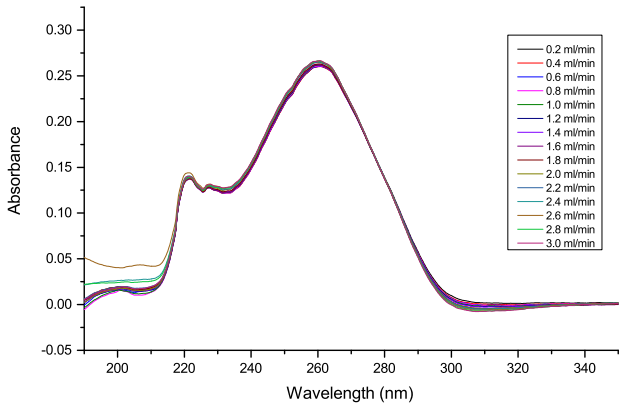
4.7.3 Extensional flow in the deep cross-slot

Figure 4.30 displays linear dichroism, absorbance and reduced LD spectra of DNA solution passed through the deep cross-slot microfluidic device. An important feature to note with this data is the positive sign of the LD peak at 260 nm in Figure 4.30a. This is due to the experimental setup, where the stretching axis of the cross-slot is *perpendicular* to the alignment direction used for the wide channel and Couette flow experiments. The absorbance values measured increase with flow rate, indicating that the deep cross-slot does suffer from a similar "bulging" problem to the deep wide channel. However, while increases due to flow rate were as much as 40% in the deep wide channel, the maximum seen with the deep cross-slot device was 5%.

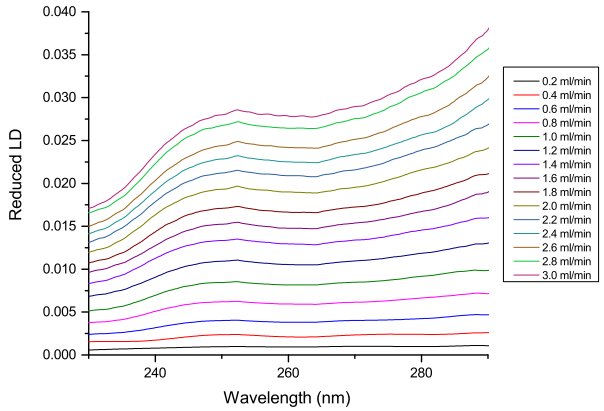
Despite this issue, the trend of an increased strength of LD and reduced LD signal at 260 nm with increased flow rate is seen here. Furthermore, the magnitude of the reduced LD signals obtained with this device are much larger than those obtained with the deep wide channel device.



(a) Linear dichroism



(b) Absorbance



(c) Reduced LD

Figure 4.30: Linear dichroism, absorbance and reduced LD spectra of DNA solution in extensional flow through the deep cross-slot microfluidic device.

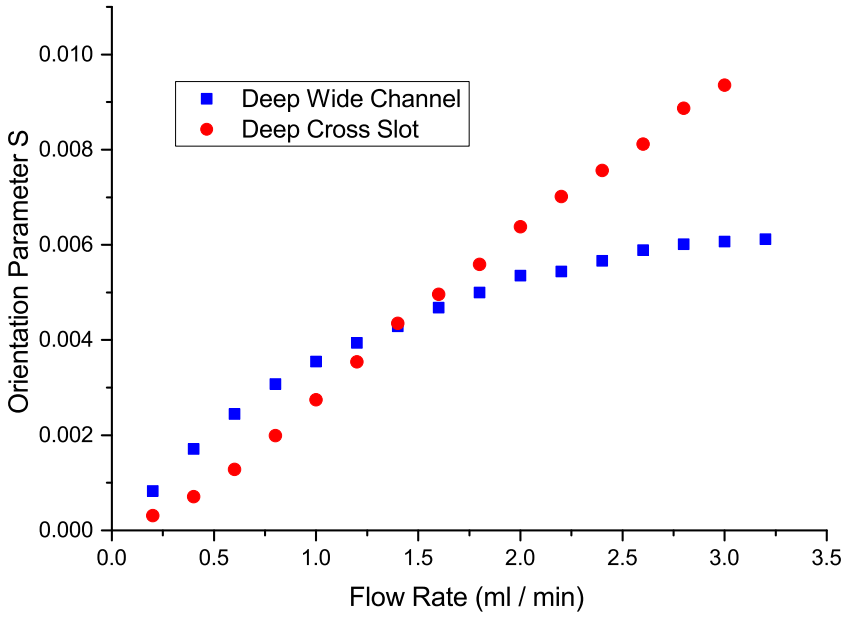
4.7.4 Comparison of orientation performance

Due to the deformation of the wide channel with increasing flow rate, the approximations of Pozrikidis [110] used for previous devices cannot be used here to provide an average shear rate for the deep wide channel. Therefore direct comparison of the shear rates is not possible in this situation. Furthermore, comparison of the cross-slot device with either the wide channel device or Couette flow cannot be done using strain rates, as it uses extensional strain rather than shear strain.

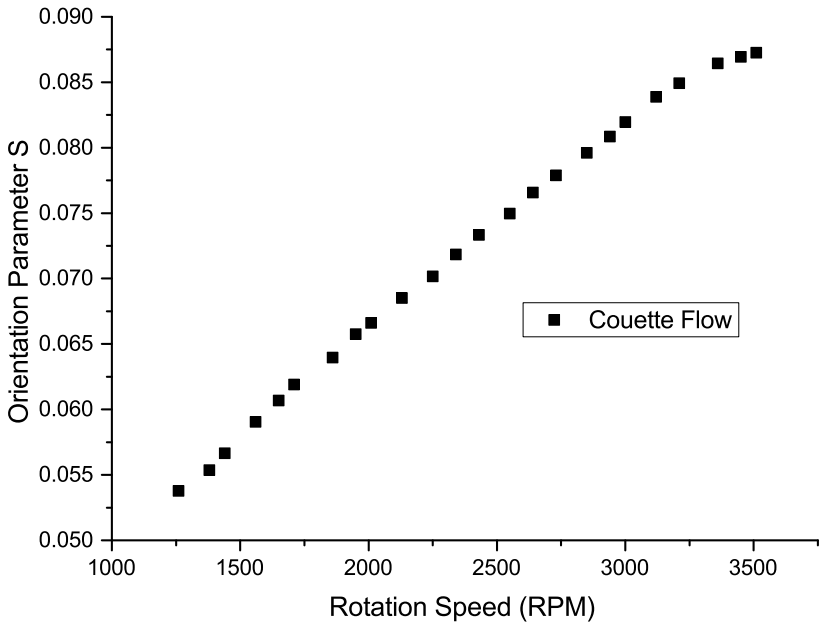
However, it is possible to calculate orientation parameter values based on the reduced LD values obtained at 260 nm in each experiment, using Equation (1.1.5) and the angle between the molecular alignment axis and the chromophores as $\alpha = 86°$, established by Norden et al. [96]. For the cross-slot, $\alpha = 4°$ is used instead, due to the stretching axis being perpendicular to the alignment axis for the other devices. Furthermore, since both microfluidic devices use the same pump, it is possible to do a practical comparison of the two devices by plotting the orientation parameters against the flow rate. Plots of the orientation parameter against flow rate (for the microfluidic devices) or rotation speed (for Couette flow) are shown in Figure 4.31.

The data in Figure 4.31a indicate that the performance of the deep cross-slot in aligning DNA molecules is significantly better than the deep wide channel at higher flow rates, although both devices are comparable at flow rates below 1.5 ml min^{-1}. The "flattening" seen with the deep wide channel is most likely due to the "bulging" phenomenon, where the channel depth increases with the pressure of the fluid passing through the channel. This means that succesive increases in the flow rate do not produce the same increases in the proportion of molecules aligned with the flow.

Future experiments with the deep-cross slot could be performed and compared with simulations of bead-spring chains in extensional flow, as discussed in Chapters 2 and 3. However, with regard to practicality, the orientation parameter values achieved by the Couette flow cell are significantly greater than both microfluidic devices. Furthermore, the sample volumes required for the microfluidic devices are of the order of millilitres, whereas only 70 µl is required for the Couette flow cell. While refinements can be made to the microfluidic setups to reduce the volumes required, this data suggests that, for polymers like DNA, Couette flow is better for aligning molecules for LD experiments.



(a) Linear dichroism



(b) Reduced LD

Figure 4.31: Linear dichroism & reduced LD spectra of DNA solution in extensional flow through the deep wide channel microfluidic device and a plot of reduced LD at 260 nm against flow rate.

4.8 Summary and future work

In this chapter, the development of simple microfluidic devices manufactured using PDMS and their testing with DNA solutions in LD experiments has been described. Initial tests with pressure-driven flows through a wide microfluidic channel with a shallow depth (approx 40 μm) demonstrated that LD experiments could be performed with such a device and increased fluid flow rates resulted in more alignment of DNA molecules, based on linear dichroism spectra measured. However, the shallow depth of the channel was thought to be causing a poor signal-to-noise ratio, due to the low path length for absorbance. The device was, therefore, refined to have a larger channel depth.

A cross-slot device, based upon a design published by Haward et al. [44], was developed to test extensional flows with DNA molecules in LD experiments. This was prepared to a similar channel depth as the revised wide channel design. Each microfluidic device was shown to be able to obtain LD data from DNA solutions. Furthermore, the deep cross-slot device was determined to be able to align a higher proportion of DNA molecules at high flow rates than the deep wide channel device. It is thought that the diminishing ability of the deep wide channel device to align at higher flow rates is due to the thin PDMS wall bulging out at high fluid pressures.

The alignment of DNA molecules using the deep cross-slot device was shown to be poorer than that obtained in the Couette flow cell. Furthermore, the sample volumes required for the microfluidic devices are two orders of magnitude greater than for Couette flow. Optimisation of the microfluidic devices to use a smaller volume might be possible in future. However, these results indicate that the Couette flow cell is more efficient in aligning DNA molecules for LD experiments than the microfluidic devices tested.

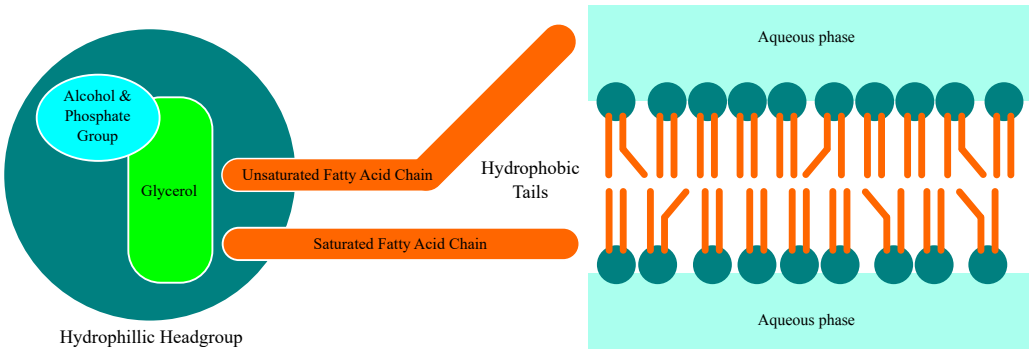
Future work with the cross-slot microfluidic device could be performed in conjunction with bead-spring chain simulations, to compare predictions of the behaviour of polymers in extensional flow with experimental observations. These experiments could be performed with fixed lengths of DNA, obtained, for example, by cutting pre-assembled plasmids in one place. The LD and derived molecular alignment measurements made with in these experiments could then be compared with calculations of predicted alignment parameter values in simulations of bead-spring chains in fluid flow.

Chapter 5

Linear dichroism experiments with vesicles in flow

In the previous chapter, linear dichroism spectroscopy experiments were performed with DNA solutions flowing through microfluidic devices. These were shown to be able to align DNA molecules sufficiently to give an LD signal. This chapter considers the use of these devices to align lipid vesicles and structures within their membranes, to enable LD experiments to be carried out. This is with a view to using them to study membrane proteins in the future. Particular focus is given to the cross-slot device developed in the previous chapter, because of theoretical and experimental evidence suggesting that the use of extensional flows is most likely to give rise to so-called "tank-treading" behaviour in vesicles. This was argued in Chapter 1 to be the most preferred vesicle behaviour in the context of membrane protein LD experiments.

This chapter details LD experiments of vesicles, with and without membrane probes, using Couette shear, pressure-driven (Poiseuille) & extensional flows. Preliminary experiments were performed with microfluidic devices developed in the work with DNA, as described in the previous chapter. Alterations to the protocol for the preparation of vesicles are discussed, along with data illustrating their effects on the size of vesicles produced.



(a) Outline structure of a phospholipid

(b) Lipid bilayer structure — two layers of phospholipids arranged with the hydrophobic tails pointing inwards.

Figure 5.1: Illustration of the basic components of phospholipids and a cross-section of a phospholipid bilayer.

5.1 Introduction — Lipids, vesicles and DPH

Phospholipids are molecules made up of four consitutent parts: an alcohol with a phosphate group, glycerol and two fatty acid chains. The acid chains are hydrophobic, whereas the rest of the molecule is hydrophilic. This means that when these molecules are packed together and exposed to aqueous environments, they will tend to form bilayers such as that illustrated in Figure 5.1. A vesicle is constitued of a volume whose boundary surface is made up of a lipid bilayer. The fluidity of a vesicle's surface bilayer is dependent on the lipids that make up the bilayer. The presence of fatty acid chains that are unsaturated will result in kinks that mean that the bilayer is more fluid, as the lipids will not be tightly packed together. This also affects the transtion temperature — the temperature at which a bilayer transitions between a packed, gel structure to a more fluid and flexible state. If there are more unsaturated fatty acid tails present, then this temperature is reduced.

In this investigation, vesicles made up of phosphatidylcholine (PC) lipids from soy bean lecithin (obtained from Sigma Aldrich®) were used. The distribution of lipids tails in this mixture is shown in Figure 5.2. The high presence of unsaturated chains means that the transition temperature is −25 °C [70]. This means that at room temperature, the insertion of membrane probe molecules for experiments is possible, as the bilayers of vesicles will not be in the relatively rigid gel phase.

Vesicles made up from molecules of a specific lipid were considered but thought not to be of practical use in this project. The volumes of vesicles required for experiments (of

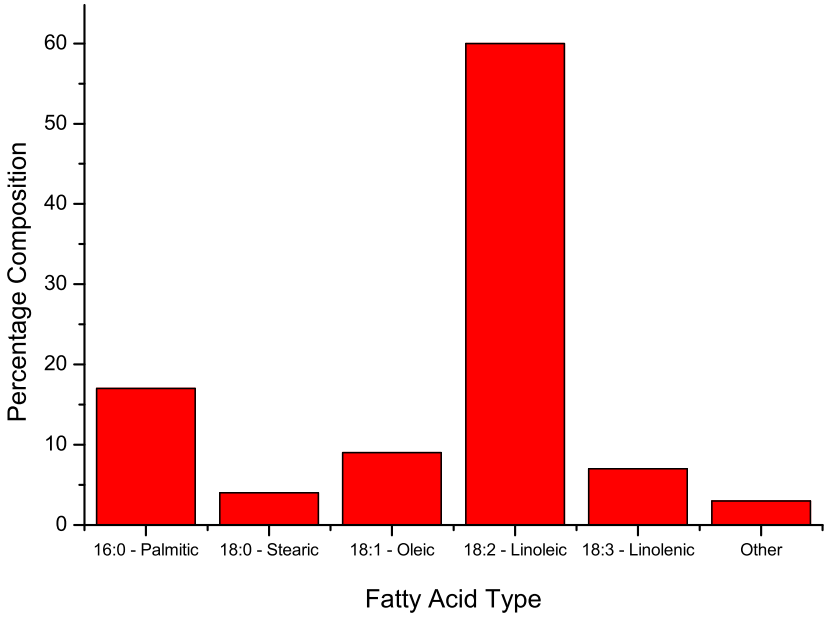


Figure 5.2: Distribution of fatty acids present in soy bean PC sourced from Sigma Aldrich® P3644. The numerical X:Y labels refer to the number of carbons in the fatty acid chain (X) and the number of unsaturated carbon double bonds present in the chain (Y).

the order of millilitres, due to accounting for the tubing and pump dead-volume) would require large amounts of purified lipid and would be an expensive undertaking. The transition temperatures of these lipids also tend to be higher as a result of their purity; thus the insertion of molecules into bilayers at room temperature would be more challenging.

As mentioned in Chapter 1, the motivation for studying vesicles with membrane probes is the structural study of membrane proteins. One technique that is widely used in this field is circular dichroism (CD), whereby the secondary structure content of membrane proteins embedded in membranes of vesicles is measured (see Section 1.4). However, CD experiments can only be performed on vesicles with diameters no greater than 400 nm [160]. This is because large particles can cause scattering and distort CD spectra measurements.

Since the study of membrane proteins is the motivation for this investigation, we use this as the target size for vesicle preparation. The small size will also help in the reduction of light scattering in LD experiments.[1] However, the small vesicles will be harder to align in fluid flows as there is a smaller object for forces to act upon.

[1] The phenomenon of polarised light scattering is being investigated in our lab as part of a separate project.

LD experiments with membrane proteins in vesicles aligned using Couette flow have been carried out and documented in the past [18, 84, 97, 116, 121]. A goal of this project is to see if extensional flow in a microfluidic device can also be used to align vesicles for LD experiments on membrane proteins. Whereas the Couette flow cell has been optimised for use in LD experiments, the microfluidic devices in this project have not been optimised for use in this setting at all. Therefore, the use of membrane proteins for such studies would be inappropriate due to the need to synthesise and purify such proteins in large amounts. Instead, 1,6-diphenylhexa-1,3,5-triene (DPH) was used. This molecule inserts itself into lipid bilayers and aligns with its long axis parallel to the fatty acid tails of the lipids. This chemical and derivatives of it have been used as membrane probes in previous studies [74, 104, 136].

The absorbance spectrum of DPH is notable for three peaks at 339 nm, 355 nm and 373 nm which correspond to long axis transitions along the chain of the molecule and peaks at 259 nm, 266 nm and 273 nm corresponding to transitions along the short axis, as displayed in Figure 5.3 [10, 32, 41, 51].

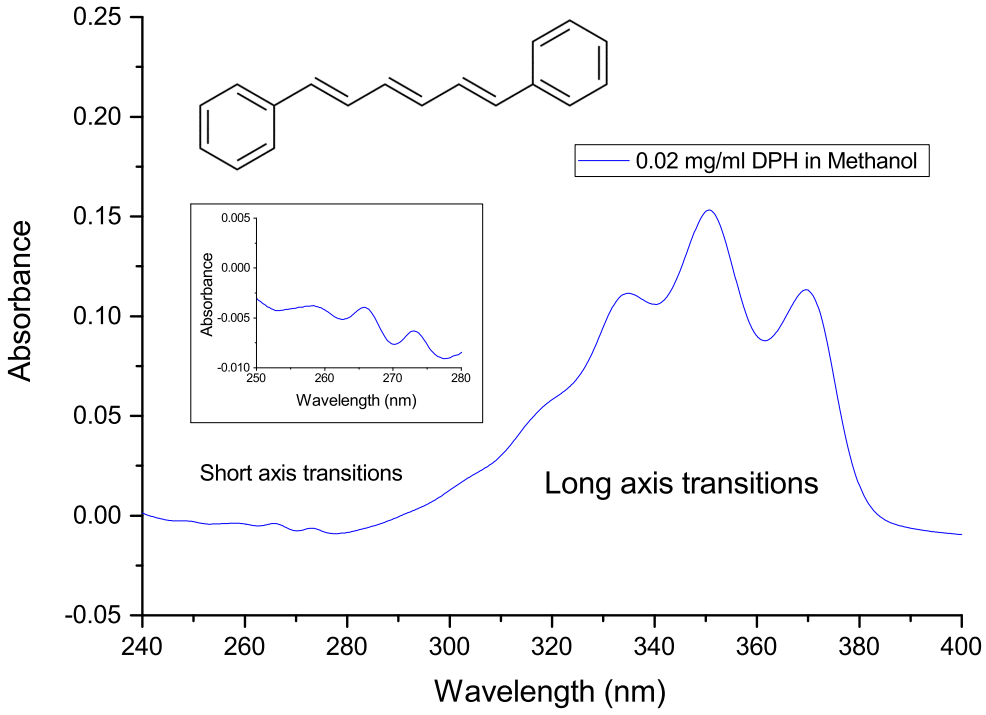


Figure 5.3: Absorbance spectrum of 1,6-diphenylhexa-1,3,5-triene (DPH) in methanol (0.02 mg ml^{-1}), with a diagram of a DPH molecule and an inset of the peaks between 250 nm and 280 nm. Peaks corresponding to long- and short-axis transitions are marked on the plot.

Assuming that DPH molecules distribute themselves evenly in the lipid bilayer of a vesicle, and the vesicles are stretched by the fluid flow, then more of the DPH molecules are expected to align with their long axes perpendicular to the long (streched) axis of the vesicles (as described in Figure 1.13 in Chapter 1). If the "parallel" polarisation in an LD measurement is taken to be along the long vesicle axis, we can therefore expect to see the peaks at 339 nm, 355 nm and 373 nm to be negative and the peaks at 259 nm, 266 nm and 273 nm to be positive.

5.2 Initial protocol for vesicle preparation

Prior to the commencement of this study, work was carried out by others in our lab to establish a protocol for the manufacture of stable vesicles with soy bean PC [79]. The protocol outlined below is for the preparation of 2 ml of soy bean PC vesicles at a concentration of 20 mg ml^{-1}, with DPH added as appropriate.

40 mg of soy bean PC was weighed into a 100 ml round bottom flask. If required, DPH was also added at this stage, with the mass depending on the final concentration desired. Approximately 2 ml of chloroform was added to the flask to dissolve the lipid and DPH. The chloroform was then removed on a rotary evaporator for 2 hours, with the water bath left unheated and rotation set to 40 rpm.

The lipid film was resuspended in 2 ml water either by gently passing the fluid over the film, or by sonication in a bath-type sonicator. While sonication is faster, there is a risk of producing vesicles that are too small if the sample is over-sonicated. The vesicles were then placed into aliquots for storage at −20 °C.

When required, vesicles would be taken out of the freezer and left to thaw in a water bath set at 25 °C. Three further cycles of freeze-thawing of the vesicles would then be carried out. Freezing would be perfomed by immersing the container with the vesicle solution into a cooling bath with dry ice and ethanol (−70 °C). Solutions would then be left in a water bath set at 25 °C to thaw. This method of applying cycles of freezing and thawing to liposome solutions was developed by Ohsawa et al. [99]. It has been reported in the literature to increase vesicle diameters from the order of tens of nanometres to microns [63, 79, 93].

The cycles are also thought to break up multilamellar vesicles (MLVs), vesicles which contain vesicles nested within them, to form medium and large-sized unilamellar vesicles (MUVs and LUVs) (see Figure 1.9) [152]. [2]

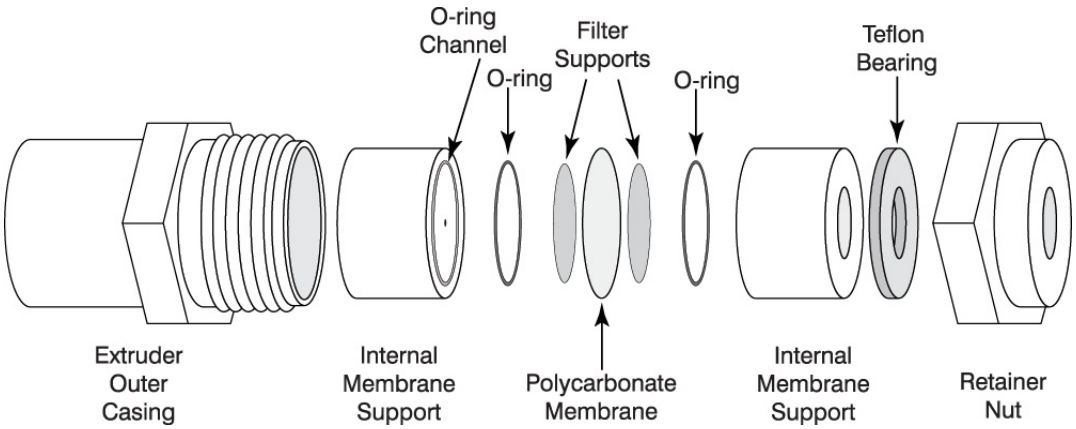
Extrusion was performed, using a Mini-Extruder Kit, to reduce the size distribution of the vesicles and ensure they are all approximately the same, reproducible size [38]. The components and construction of the extruder are given in Figure 5.4. All the components of the kit were manufactured by Avanti Polar Lipids, with the exception of the polycarbonate membranes (19 mm diameter) and filter supports (these were manufactured by Whatman). The process involves using two modified Hamilton Gastight syringes to pass a vesicle solution an odd number of times through a path-etched polycarbonate membrane with pores of a particular diameter. Initially, this protocol used membranes with 100 nm diameter pores and the fluid was passed through 11 times. Detailed instructions on the use and construction of the Mini-Extruder may be found on the Avanti Polar Lipids website [77].

Vesicle sizes were measured with dynamic light scattering (DLS), using a Malvern Instruments Zeta Nanosizer. This measures the intensity of light scattered on a sample, comparing shifts in the intensity over time. These shifts are due to the random motion of particles in the sample. Smaller particles will be responsible for larger shifts, as they are more likely to move greater distances in a short space of time than large particles, resulting in more changes to the pattern of scattered light. The instrument uses this to calculate the distribution of particle sizes present in a sample. The mathematics behind this are explained in detail in work by Schärtl [129].

Furthermore, the software can estimate the number of particles present, by noting that the intensity of scattering of a particle is proportional to the sixth power of its diameter according to Rayleigh scattering theory [134]. However this number data is based on calculations on the measured intensity data, and the errors may arise due to the sensitivity of the calculation for smaller particles and if there are small errors in the intensity measurements in that size region [52].

[2] The effect of shear flows on MLVs has been studied previously, demonstrating that nested vesicles inside an MLV can exhibit a range of behaviours, depending upon their sizes [109]. In particular, the alignment of the external vesicle need not result in the alignment of nested vesicles. Therefore multilamellar vesicles should be avoided in LD experiments.

(a) Photo of the whole extrusion assembly, including Gastight syringes.



(b) Schematic of the composition of the extruder.

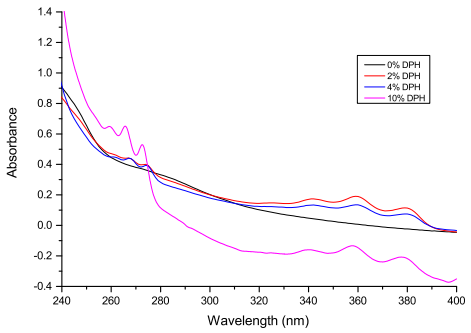
(c) Photo of the extruder components and a filter support.

(d) Photo of a polycarbonate membrane and the extruder outer casing.

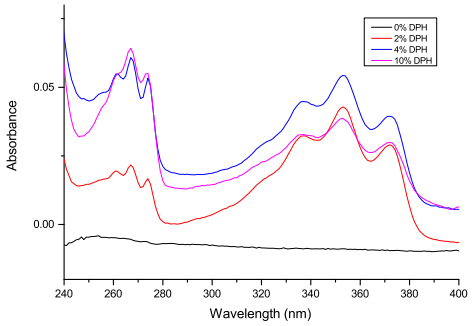
Figure 5.4: Diagram and photos of the Avanti Polar Lipids Mini-Extruder Kit used to extrude lipid vesicle solutions.

5.3 Experiments with Couette flow and the wide channel microfluidic device

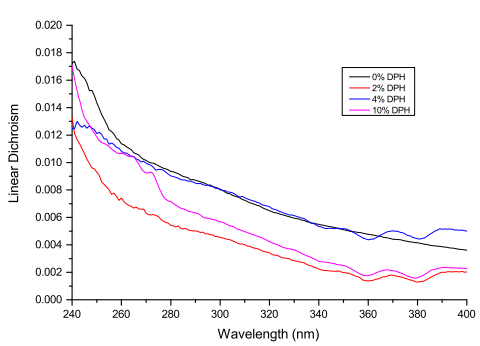
To establish the quality of the absorbance and LD signals that could be obtained from vesicles with DPH in microfluidic channels made with PDMS, preliminary experiments were performed using the wide channel microfluidic device as described in Section 4.4. The vesicles were manufactured using the protocol above with DPH added to lipid mixture as a proportion of the total mass of soy bean PC and DPH. Spectroscopic measurements were also taken with Couette flow (where the flow measurement was taken with the cuvette rotating at 3000 rpm) for comparison. In addition, the size distribution of vesicles were measured using DLS.



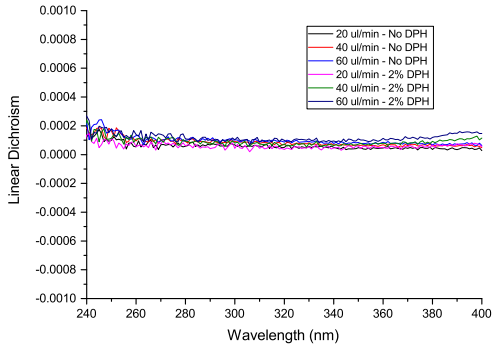
(a) Couette flow absorbance



(b) Pressure-driven flow absorbance



(c) Couette flow LD



(d) Pressure-driven flow LD

Figure 5.5: Absorbance and LD spectra of soy bean PC vesicles containing DPH (0%, 2%, 4% and 10% by mass) in Couette shear flow and pressure-driven flow through the wide channel microfluidic device. The LD spectra of soy bean PC vesicles with 4% and 10% DPH by mass in pressure-driven flow are in the Appendices in Figure F.1

The Couette flow absorbance spectra in Figure 5.5a display the peaks that come from the

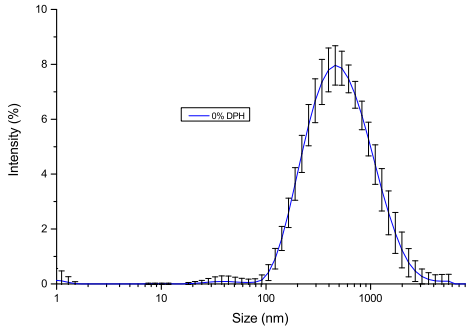
long-axis transitions in membrane-bound DPH molecules; the LD spectra in Figure 5.5c show that these signals come from DPH molecules with long axes perpendicular to the flow. The absorbance spectra for both the Couette and pressure-driven flows (Figures 5.5a and 5.5b) slope upwards due to light scattering by the vesicles [97]. However this scattering background is not consistent between experiments, partly due to differences in the vesicle-size distribution and particle size strongly influences the level of scattering [97].

Despite being unable to subtract a scattering background from the data for this reason, it is possible to see that the sizes of peaks related to DPH are not consistent with the amount of DPH present (indicated by mass proportion of lipid and DPH mixture). This is unexpected, as absorbance is proportional to concentration according to the Beer-Lambert law (Equation (1.1.2)). Therefore, this suggests that addition of DPH, even at a mass concentration of 2%, is high enough for the molecules to be embedded into the vesicle membranes. Further evidence of this was found in the extrusion process in vesicle preparation, where the membranes were yellow in colour after use, due to the presence of excess DPH.

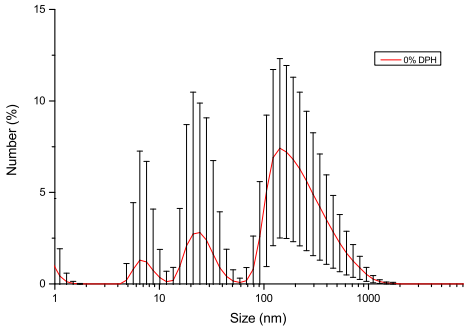
The LD spectra obtained for vesicles in pressure-driven flow through the wide channel device (Figures 5.5d and F.1) do not indicate any features beyond noise around zero. Assuming that DPH molecules are evenly distributed in a vesicle's membrane, a zero LD signal indicates that DPH molecules are isotropically oriented in the sample. This suggests that the fluid flow in the wide channel device is not able to elongate the vesicles.

The flow rates were limited by the risks of leaks in the system; thus testing at rates of around $100\,\mu l\,min^{-1}$ could not be carried out. However, the flow rates of $20\,\mu l\,min^{-1}$, $40\,\mu l\,min^{-1}$ and $60\,\mu l\,min^{-1}$ correspond with mean shear rates of $144\,s^{-1}$, $287\,s^{-1}$ and $431\,s^{-1}$ (based on the calculations referred to in Subsection 4.4.3). This is in comparison with the $1000\,s^{-1}$ shear rate in the Couette flow cell. The lack of any signal in the pressure-driven flow is therefore puzzling. One explanation could be that in Couette flow, the shear force is roughly constant across the gap between the stationary rod in the centre and the rotating cuvette, whereas the greatest shear effect in the wide channel is close to the walls and not in the centre of the flow. It is likely, therefore, that most vesicles that flow through the device are not exposed to the high shear rates required to stretch them.

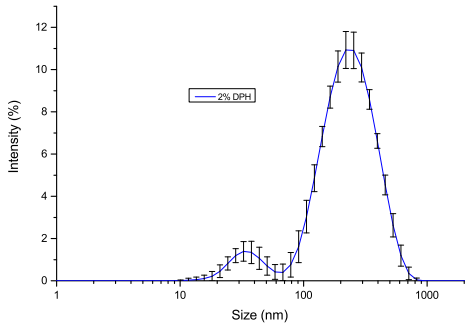
The size of vesicles is also a factor, as smaller vesicles are less able to deform. The



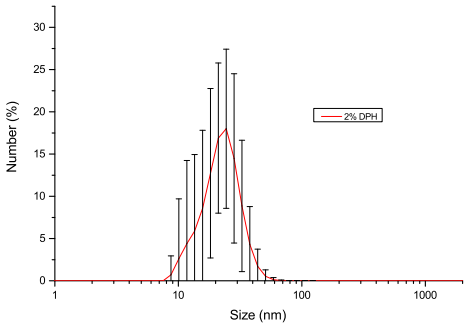
(a) Size by intensity, 0% DPH by mass.



(b) Size by number, 0% DPH by mass.



(c) Size by intensity, 2% DPH by mass.



(d) Size by number, 2% DPH by mass.

Figure 5.6: Vesicle size distribution data obtained using DLS for soy bean PC vesicles containing 0% and 2% DPH by mass before use in flow experiments.

vesicle size distribution data shown in Figures 5.6, F.2 and F.3 indicate that, according to intensity, the vesicles with DPH produced by the extrusion process were mostly between 100 nm and 600 nm in diameter, with median diameters measured to be approximately 200 nm. However, vesicles manufactured without DPH gave rise to a much wider distribution of sizes, with a median peak at around 500 nm. Furthermore, for vesicles with and without DPH, intensity peaks were observed for vesicles with diameters between 20 nm and 70 nm. These peaks resulted in the calculated number distributions indicating most vesicles present in the samples to be smaller than the target diameter of 100 nm.

5.4 Experiments with a shallow cross-slot microfluidic device

The results of experiments with the wide channel device indicated that a pressure-driven flow would not provide the stretching and alignment of vesicles that Couette shear flow does. However, the absorbance data obtained from the microfluidic setup indicate that, even though the signals are an order of magnitude smaller than those obtained with Couette flow, the quality of the signal is much cleaner and less affected by the sloping scattering background. So this suggested that microfluidics could still be a viable setup for conducting LD experiments with vesicles, but would require a flow setup that would result in alignment.

In Chapter 1, it was postulated that extensional flows would be best for linear dichroism studies with vesicles because, in addition to the act of elongating the vesicles to cause an anisotropic distribtuion of orientations of membrane bound molecules, this type of flow was least likely to cause either of the trembling or tumbling motions. These motions were considered unlikely to result in a system where an LD signal would be easily detected.

In order to create an extensional flow to stretch vesicles, we need to overcome the inertial rotational forces of the fluid that are acting on the vesicle. The rotational Péclet number of an object is a ratio of the strength of hydrodynamic interactions against Brownian forces. It is defined as the ratio of hydrodynamic shear flow and the rotational diffusion constant [146]. For a sphere, the rotational Péclet number, α, is given by:

$$\alpha = \frac{8\pi R^3 \frac{\mathrm{d}v_x}{\mathrm{d}x}\eta}{k_B T}$$

where

- R is the radius of the vesicle.
- $\frac{\mathrm{d}v_x}{\mathrm{d}x}$ is the strain rate.
- η is the dynamic fluid viscosity.

A situation with $\alpha \geq 10$ will ensure that the inertial rotational forces can be overcome by the fluid flow. Assume that:

- $R = 10^{-7}\,\mathrm{m} = 100\,\mathrm{nm}$
- $\eta = 8.9 \times 10^{-4}\,\mathrm{kg\,m^{-1}\,s^{-1}}$ (water at 25 °C).
- $k_B T = 4.11 \times 10^{-21}\,\mathrm{J}$

Then, to obtain $\alpha \geq 10$:

$$\frac{\mathrm{d}v_x}{\mathrm{d}x} \geq \frac{10 k_B T}{8\pi R^3 \eta} = 1837.5\,\mathrm{s^{-1}}$$

Now

$$\frac{\mathrm{d}v_x}{\mathrm{d}x} \approx \frac{U}{H} =: \frac{\text{Fluid velocity}}{\text{Half of central cross chamber width}},$$

and

$$U \approx \frac{\text{Flow rate}}{\text{Channel width} \times \text{Channel depth}}$$

Therefore, to get $\alpha \geq 10$,

$$1837.5\,\mathrm{s^{-1}} \geq \frac{\text{Flow rate}}{H \times \text{Channel width} \times \text{Channel depth}},$$

For the extensional cross-slot used in these experiments (shown in the previous chapter):

$$\text{Channel width} = 0.4\,\mathrm{mm}$$

$$\text{Half of the width of central chamber} = H = 1\,\mathrm{mm}.$$

Thus,

$$\frac{\text{Flow rate}}{\text{Channel depth}} \geq 1837.5\,\mathrm{s^{-1}} \times 4 \times 10^{-7}\,\mathrm{m^2} = 7.5 \times 10^{-4}\,\mathrm{m^2\,s^{-1}}$$

Table 5.1 displays the minimum flow rates required to satisfy this condition for different channel depths. It was recognised, however, that the flow rates in Table 5.1 would be too high for use in the devices; the pressure in the system would cause leaks or the pump

Table 5.1: Minimum flow rates required for flow in the cross-slot design to overcome Brownian rotational forces, on the assumption of treating vesicles as spheres.

Channel depth		Minimum flow rate	
in μm	in m	in $m^3\,s^{-1}$	in $ml\,min^{-1}$
50	5×10^{-5}	3.75×10^{-8}	2.25
100	1×10^{-4}	7.5×10^{-8}	4.50
200	2×10^{-4}	15×10^{-8}	9.00
500	5×10^{-4}	3.75×10^{-7}	22.5

would not be able to push fluid through the device at the flow rate required. This meant that the flow rates used in experiments were of an order of magnitude lower than those given in the table; this also corresponds with Péclet numbers of the order of 1. Experiments with the wide channel device also indicated that the absorbance and LD intensities measured were much lower than those measured with the Couette flow device. One possible explanation for this was the path length being too low. With this in mind, a cross-slot device was prepared with a depth of roughly 100 μm.

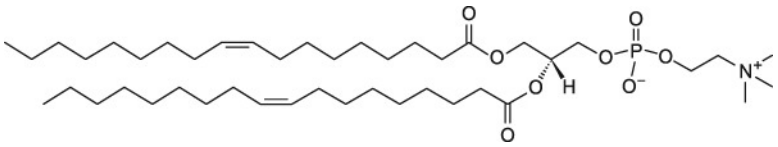
To overcome the problem of excess DPH being used in previous vesicle preparations, a base quantity of DPH was selected. The working assumption was that there should be 1 molecule of DPH for every 100 lipid molecules. Using the distribution of lipids present in soy bean PC (see Figure 5.2), a number of example PC lipids were chosen with tails present in the soy bean PC distribution. These included DOPC (18:1), POPC (16:0-18:1) and PLPC (16:0-18:2) with respective molecular weights 786.113 $g\,mol^{-1}$, 760.076 $g\,mol^{-1}$ and 758.060 $g\,mol^{-1}$. Using DOPC as an example,

$$\begin{aligned} \text{DPH mass}: &\quad 255.355\,g\,mol^{-1} \times 1\,mol = 255.355\,g \\ \text{DOPC mass}: &\quad 786.113\,g\,mol^{-1} \times 100\,mol = 78\,611.3\,g. \end{aligned}$$

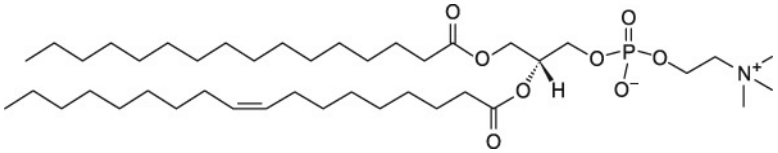
This gives

$$\text{DPH mass concentration} = \frac{255.355}{255.355 + 78611.3} = 0.00323781 = 0.32\%$$

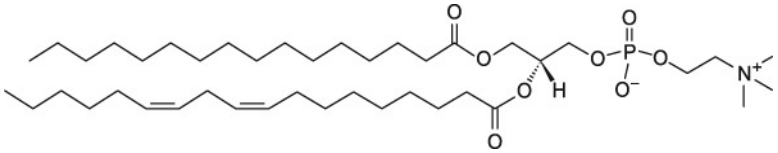
Similar values were obtained with POPC and PLPC. To offset losses due to the vesicle preparation process, a higher figure of 0.4% was chosen to be the mass proportion that DPH would be added in future.



(a) DOPC — 1,2-dioleoyl-sn-glycero-3-phosphocholine



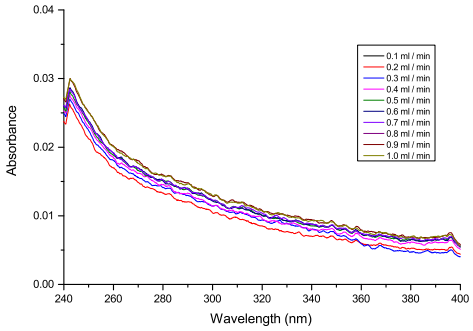
(b) POPC — 1-palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine



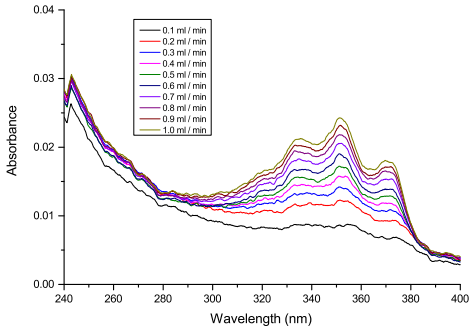
(c) PLPC — 1-palmitoyl-2-linoleoyl-sn-glycero-3-phosphocholine

Figure 5.7: Structures of three PC lipids with chains typical to those found in soy bean PC.

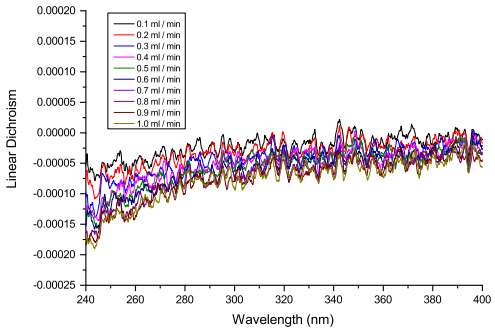
Finally, as a means to stabilise vesicles and prevent breakdown into smaller vesicles during extrusion or fluid flow, vesicles were prepared in buffer solution instead of water. A 10 mM pH7.4 Sodium Phosphate buffer was chosen, prepared using protocols in Russell and Sambrook [125].



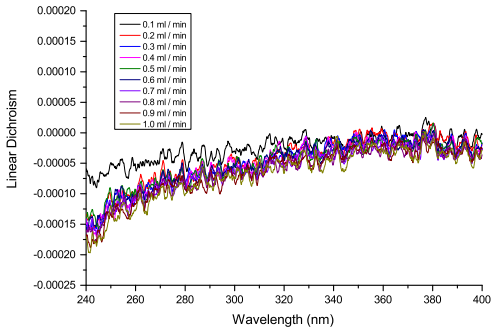
(a) Absorbance, No DPH



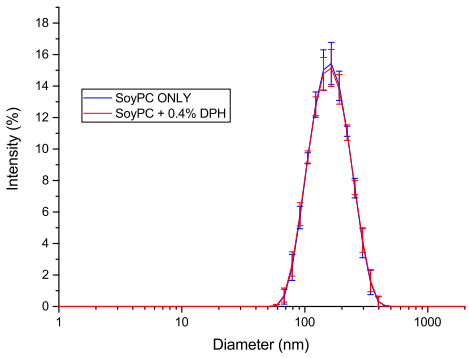
(b) Absorbance, With DPH



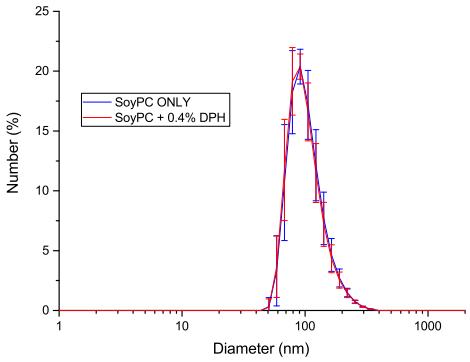
(c) Linear Dichroism, No DPH



(d) Linear Dichroism, With DPH



(e) Size by intensity for vesicles with no DPH.



(f) Size by number for vesicles with no DPH by mass.

Figure 5.8: Absorbance and LD spectra of soy bean PC vesicles without DPH and with DPH (0.4% by mass) in extensional flow in the cross-slot microfluidic device, and vesicle size distribution data obtained using DLS for soy bean PC vesicles containing no DPH before use in measurements using the cross-slot device.

Figures 5.8e and 5.8f show that the vesicle diameters, both when measured using scattering intensity and with derived number distribution, were spread between 60 nm and 300 nm. This applied for vesicles with and without DPH, indicating that the presence of the membrane probe did not cause noticeable changes to the vesicle size distribtion.

However, the LD spectra in Figures 5.8c and 5.8d show no DPH peaks. This is despite the presence of DPH peaks in the absorbance spectra, shown in Figures 5.8a and 5.8b. The increasing size of the DPH absorbance signal with flow rate also indicate that the device experienced the same path length issues as the deep wide channel when tested with Potassium Chromate solution in Subsection 4.6.3.

5.5 Vesicle preparation and method development

The preliminary vesicle experiments showed that absorbance spectra with peaks from long axis electronic transitions in DPH molecules, but only Couette flow could yield a reasonable LD signal. Work was carried to investigate possible factors for this. The channels in both devices were made deeper, as described in the previous chapter, to increase the path length, increase the absorbance signal and, potentially, increase the strength of the LD signal to overcome the noise. The use of sodium phosphate buffers instead of water, as described above, also had an effect in preventing the formation of vesicles with diameters smaller than 50 nm.

Other variables were also considered, including the method of vesicle manufacture using freeze-thaw cycles and extrusion using membranes with pore sizes of different diameters, the effects of pumping the solution through the microfludic devices, and the use of pure lipids instead of soy bean PC. The details of these investigations are given in Appendix E.

Overall, the following conclusions were drawn regarding the vesicle preparation:

- Soy bean PC lipid should still be used instead of a pure lipid, such as DOPC.
- 10 mM pH7.4 sodium phosphate buffer should be used instead of water to redissolve the lipid film lining the round-bottomed flask after the rotary evaporation stage.
- The vesicle solution concentration was reduced from 20 mg ml^{-1} to 10 mg ml^{-1} to reduce the pressure in the extrusion process.
- Vesicle solutions should still subjected to one slow and three fast freeze-thaw cycles. Additional cycles did not appear to affect the vesicle size distribution noticeably.
- Vesicle soultions should be passed through membranes with 400 nm diameter pores, to allow for vesicles to be large enough to be more easily oriented in fluid flows for LD experiments.

5.6 Tests with pressure-driven and extensional flows with the deep wide channel and cross-slot devices

As a final test to see whether the microfluidic devices could be used to obtain LD signals from DPH molecules in vesicle memrbanes using pressure-driven and extensional flows, lipid vesicles were prepared using the protocol below and pumped through the deep wide channel and deep cross-slot microfluidic devices, as described in Subsection 4.6.2.

5.6.1 Protocol for vesicle preparation

A 5 ml solution of 10 mg ml^{-1} soy bean PC vesicles in 10 mM pH7.4 sodium phosphate buffer was prepared using the rotary evaporator, as described in the original protocol. A second solution was prepared but including DPH at a mass concentration of 0.4%.

Each solution was placed in two large Eppendorfs (capacity 2.5 ml). These were put through one slow freeze-thaw cycle and three fast freeze-thaw cycles. The solutions were then extruded 11 times through a membrane with 800 nm diameter pores and then extruded 11 times thorugh a membrane with 400 nm diameter pores.

The reason for the change to the extrusion membranes used was to increase the size of the vesicles produced, so as to enable the fluid flows to be able to deform and orient the vesicles more easily. In particular, for the extensional flow case, doubling the radius of the vesicles being oriented means that the flow rate required to overcome inertial rotational forces is an eighth of what it was previously.

A measurement of the vesicle-size distribution was taken using DLS before using vesicle solutions in LD spectroscopy experiments. A total of 25 readings were taken for each solution. In addition, three 70 µl samples of each solution were used in LD experiments using Couette flow to orient the vesicles. This was performed to ascertain if the vesicles had taken up DPH and could be oriented in shear flow.

After a vesicle solution was used with a microfluidic device, samples were taken for size measurements using DLS and LD experiments using Couette flow. This was done to check if the sizes had changed (for example, if the vesicles were broken down by the fluid

flows) and to see if the vesicles would still produce an LD signal if oriented in Couette flow.

In the following subsections, for brevity, only data for vesicles containing DPH are given. Related data for vesicles without DPH are provided in Appendices F.2 and F.3.

5.6.2 Pressure-driven flows through the deep wide channel device

The LD spectra of soy bean PC vesicles with DPH in pressure-driven flow through the deep wide channel device are shown in Figure 5.9a. The presence of waveforms due to the HPLC pump can be seen here, as well as in the LD data for vesicles without DPH (Figure F.4a). Removal of these patterns using a fast Fourier transform was attempted, as was performed with data from experiments with DNA solutions. However, the shape of the plots combined with the smoothing technique appeared to form new peaks in the data in an inconsistent manner. The data here are therefore presented in raw form.

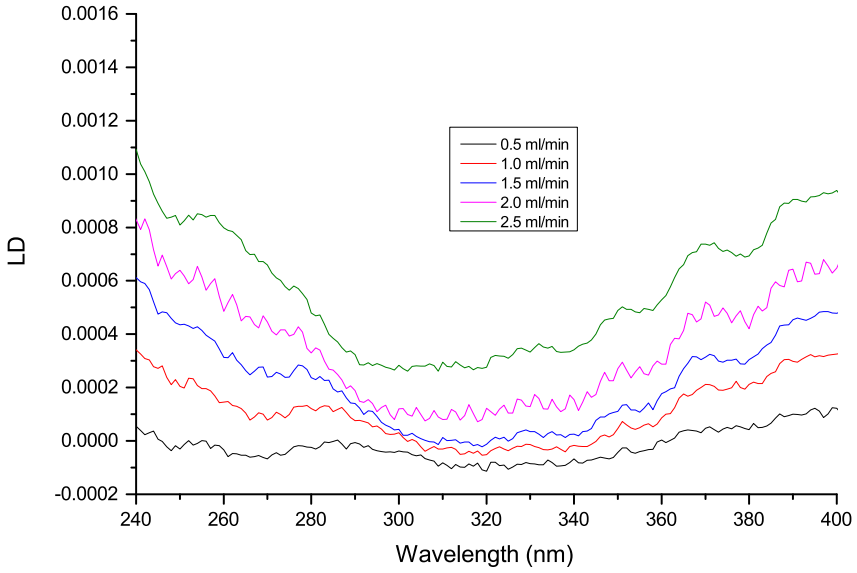
Despite the interference patterns present in the data, there appear to be negative LD "peaks" at 340 nm, 360 nm and 380 nm corresponding to the long-axis transitions of DPH. These are more noticeable in the data for flow rates of $1.5\,\mathrm{ml\,min^{-1}}$, $2.0\,\mathrm{ml\,min^{-1}}$ and $2.5\,\mathrm{ml\,min^{-1}}$. This suggests that the pressure-driven flow did result in the elongation and alignment of a proportion of vesicles in the microfluidic device.

The associated absorbance spectra in Figure 5.9b also display the peaks corresponding to the long-axis transitions of DPH. The pump interference pattern can also be seen in the plot for the $2.0\,\mathrm{ml\,min^{-1}}$ flow rate. The "bulging" effect of high flow rates through the deep wide channel device previously observed is also apparent in the data here, with increases in the absorbance value attained as the flow rate rises.

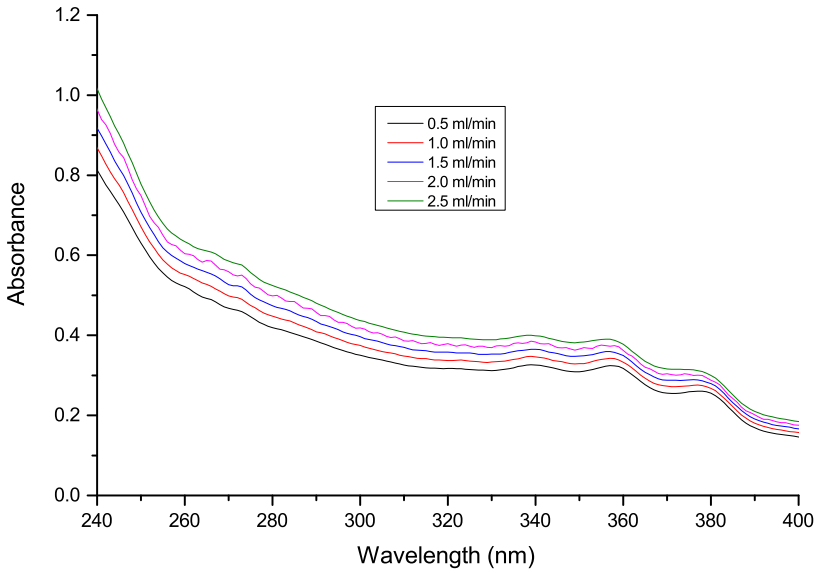
Reduced LD spectra have not been plotted here, due to the inteference patterns present in the LD data. However, the values obtained in the LD measurements are three orders-of-magnitude lower than the absorbance value. In comparison to this, the values obtained for absorbance with the vesicles in Couette flow are two orders-of-magnitude greater than the values recorded for LD, as shown in Figure 5.10. This suggests that the orientation is ten times better in Couette flow than in the pressure-driven flow used here.

The Couette flow absorbance data in Figure 5.10b also indicate a drop in the concentration

of DPH present in the vesicles after use in the microfluidic device. This is also reflected in the decreased sizes of LD signals associated with DPH measured in Couette flow LD of vesicles before and after use in microfluidics (Figure 5.10a). The size of the sloping background signal, a phenomenon attributed to light scattering, is also reduced before and after the vesicles have passed through the microfluidic device. This suggests that the vesicles were broken up into smaller bodies as a consequence. However, the size distribution data from DLS measurements, shown in Figure 5.11, indicate very little change in vesicle sizes before and after use in the microfluidic device. This was also observed with vesicles containing no DPH (see Figures F.5 and F.6).

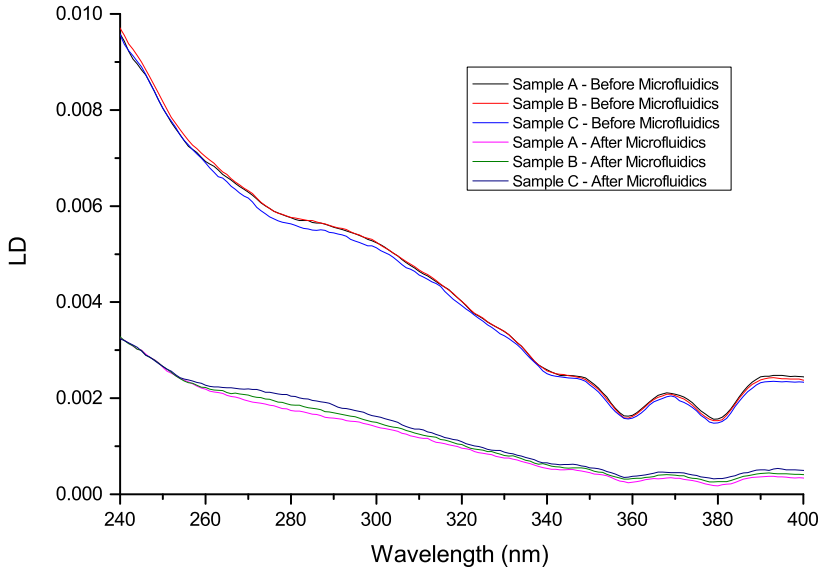


(a) Linear dichroism

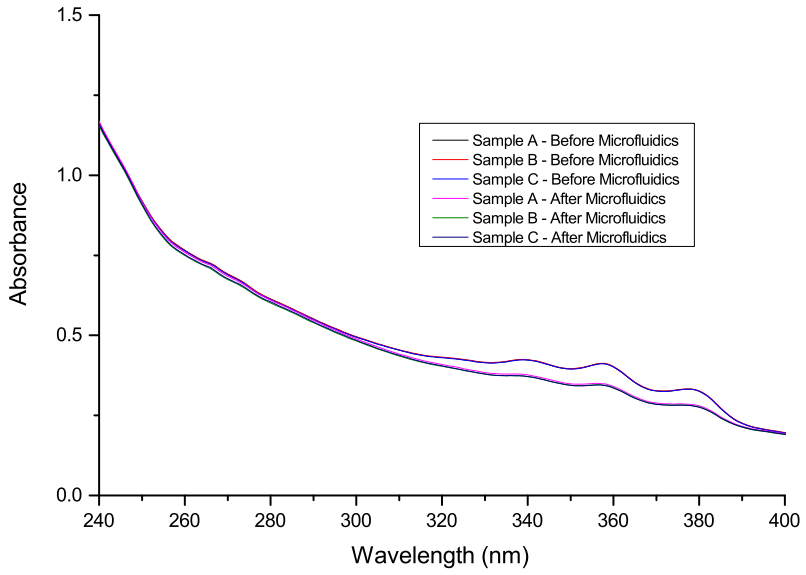


(b) Absorbance

Figure 5.9: Linear dichroism and absorbance spectra of soy bean PC vesicles with 0.4% DPH by mass, in pressure-driven flow through the deep wide channel microfluidic device at different flow rates.



(a) Linear dichroism



(b) Absorbance

Figure 5.10: Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep wide channel microfluidic device.

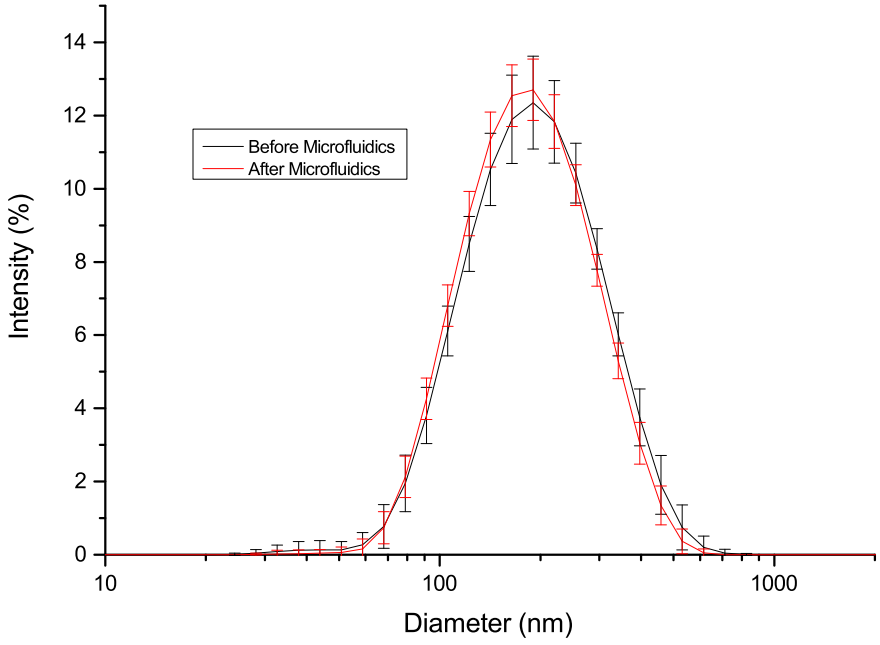


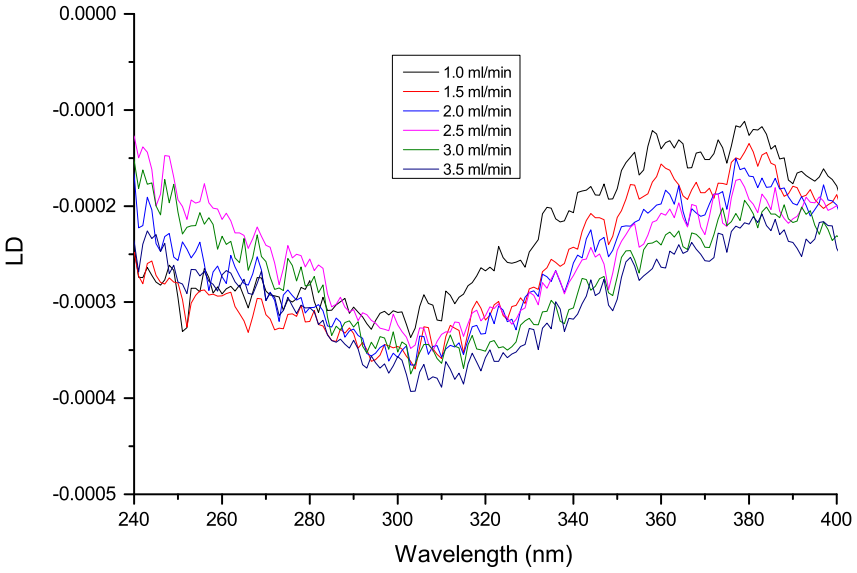
Figure 5.11: Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep wide channel microfluidic device.

5.6.3 Extensional flows through the deep cross-slot device

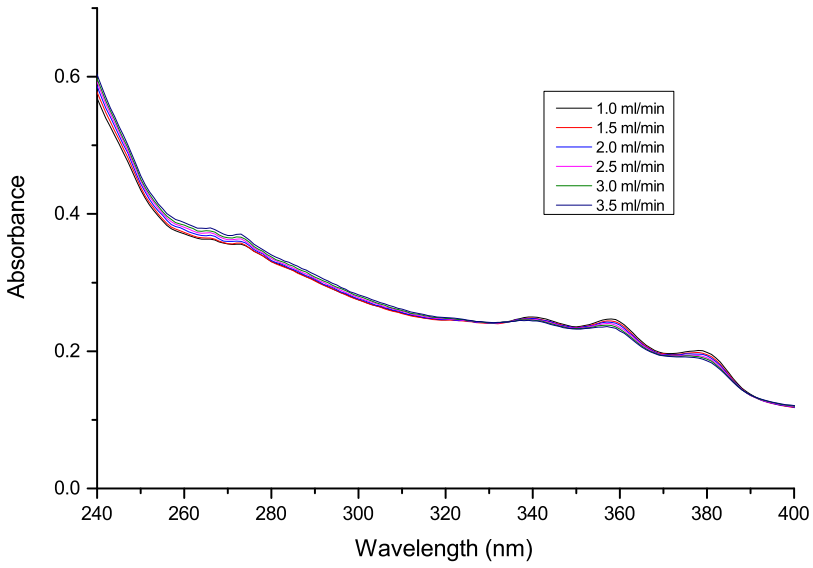
The LD spectra of soy bean PC vesicles with DPH in extensional flow through the deep cross-slot device are shown in Figure 5.12a. As with the pressure-driven flow data, the LD spectra presented here are raw. While the interference from the HPLC pump is contained in the data, it is much harder to see here; the traces are more visible in the LD data for vesicles without DPH in Figure F.7a. As with experiments using DNA solutions, the orientation of the cross-slot device with respect to the light beam means that the long-axis transitions of DPH molecules in the membranes of elongated vesicles should give a positive LD signal. However, the noise and interference present here make it difficult to determine the presence of any signal from DPH.

The associated absorbance spectra in Figure 5.12b display the peaks at 340 nm, 360 nm and 380 nm corresponding to the long-axis transitions of DPH. It is also noteworthy that there are three orders-of-magnitude difference between measurements of LD and absorbance, which suggests that, at best, there was weak alignment of vesicles in extensional flow. Therefore, either the LD data is too noisy to distinguish signals from DPH, or the LD data is just noise and the cross-slot has not been able to align vesicles to give an LD signal. As with the deep wide channel data in the previous subsection, the "bulging" effect of high flow rates through the deep wide channel device is also apparent in the cross-slot device, with increasing absorbance values attained with rises in the flow rate (Figure 5.12b). However, the effect is not as prominent here.

The Couette flow data in Figure 5.13 also indicate a drop in the concentration of DPH present in the vesicles after use in the microfluidic device. The size of the scattering background is also reduced before and after the vesicles have passed through the microfluidic device (Figure 5.13a). However, the difference in the scattering backgrounds before and after use in the deep cross-slot device is less prominent than with the deep wide channel device. This suggests that the overall size of vesicles after use in the cross-slot device was smaller; however, the decrease in size after use was more pronounced with vesicles passed through the wide channel device. Furthermore, the size distribution data from DLS measurements, shown in Figure 5.14, indicate very little change in vesicle sizes before and after use in the microfluidic device. This was also observed with vesicles containing no DPH (see Figures F.8 and F.9).

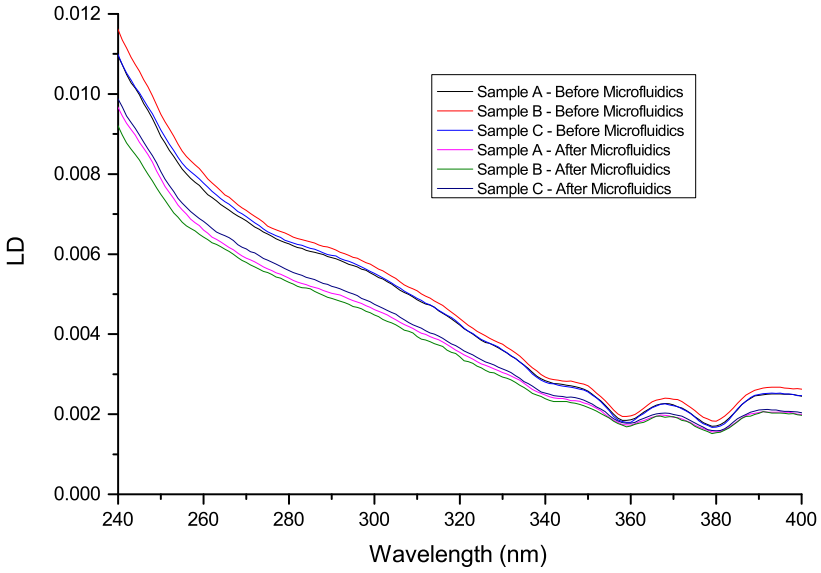


(a) Linear dichroism

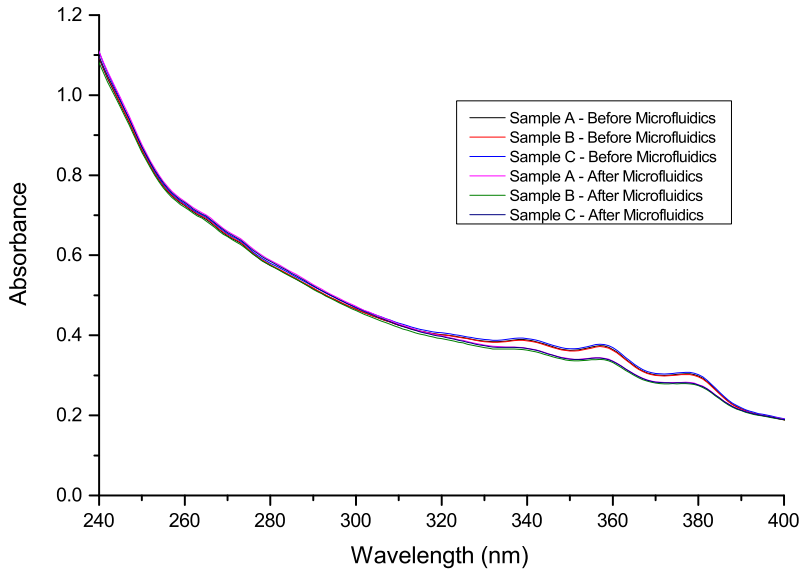


(b) Absorbance

Figure 5.12: Linear dichroism and absorbance spectra of soy bean PC vesicles with 0.4% DPH by mass, in extensional flow through the deep cross-slot microfluidic device at different flow rates.



(a) Linear Dichroism



(b) Absorbance

Figure 5.13: Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep cross-slot microfluidic device.

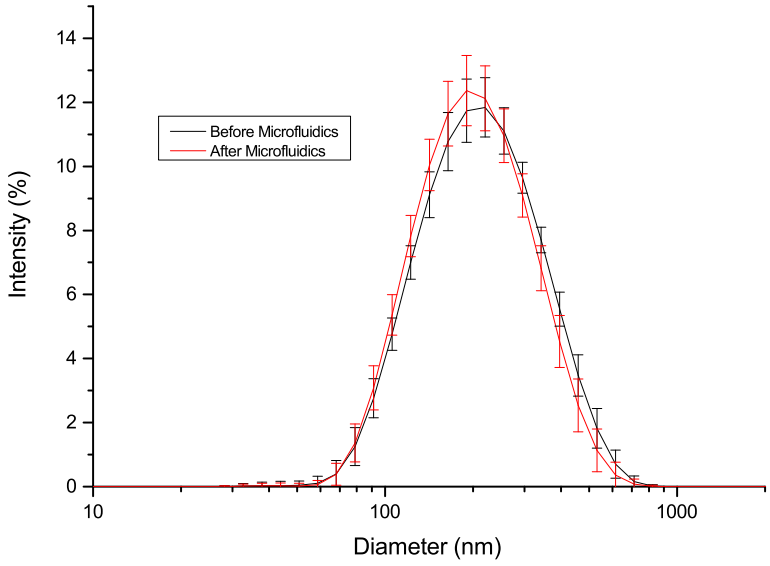


Figure 5.14: Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles with 0.4% DPH by mass, before and after use in the deep cross-slot microfluidic device.

5.6.4 Discussion

The observations made with the deep wide channel device shows that pressure-driven flows through a microfluidic device can be used to obtain LD data from molecules embedded in vesicle membranes. However, the level of orientation acheived was far lower than that obtained in Couette flow. One possible explanation for the lack of orientation is the "bulging" of the channel under high pressure fluid flows. This is a disadvantage to the use of PDMS for these applications, and indicates that the channel would need to be constructed out of more rigid materials. In the course of this project, one study by Matsuo et al. [84] used a device constructed from quartz, similar to the wide channel device investigated here, to perform LD experiments on membrane proteins. While they reported being able to obtain LD data to analyse protein structure with this device, the experimental setup required long measurement times (of the order of hours) and multiple accumulations to record a signal large enough to be distinguishable from noise. This also hints at the minimal vesicle alignment that the pressure-driven flow can provide.

With regard to extensional flow, an aim of this project was to assess whether it could be used to stretch and align vesicles to obtain LD data from molecules embedded in the membranes. This was based on the theory described in Chapter 1, which indicated that extensional flows could stretch vesicles and cause them to undergo tank-treading, and not tumbling or trembling. This was also in contrast with planar shear flows where the latter two behaviours could be observed and would not be desirable for LD experiments. The data obtained from the deep cross-slot device does not allow us to conclude if extensional flow is suitable or unsuitable for this purpose, as it was unclear whether a DPH signal was present but hidden in noise and pump interference in the LD spectra measured, or no alignment took place and so no LD signal was recorded from DPH.

There are a number of factors that would need to be addressed to investigate this further. A major factor is the difficulty of maintaining even rates of fluid flow into the central chamber of the deep cross-slot device. The formation and trapping of bubbles in the system can cause fluid flows to be blocked and the flow in the centre to not be extensional. The relatively large channel depths of the device also make it harder to remove bubbles trapped in the device. Furthermore, the use of one pump and a junction to split fluid flow evenly is not optimal. The ideal setup would involve the use of two pumps, with smooth

pumping actions, to ensure that the extensional flow is the central chamber is balanced and maintained throughout the duration of an experiment. This would also help to remove the interference patterns seen in the LD and absorbance spectra due to the HPLC pump.

A remodelled cross-slot device may also require a redesign of the central chamber. The design used here is based on work by Haward et al. [44], who developed the design to maximise the volume of fluid subject to extensional strain. However, Galindo-Rosales et al. [39] have commented that this is based on a 2D flow approximation that does not translate to PDMS microfluidic devices, where end-walls distort the flow field and the device is not able to generate a uniform well-defined extension rate inside the cross-chamber. New designs have been proposed by them instead that could be considered for the central chamber.

The use of quartz rather than PDMS should also be considered for a remodelled cross-slot device, though this would require quartz etching which is expensive. An advantage of PDMS over quartz is the relative ease with which devices can be constructed from it. However, a further problem with PDMS in the context of studies with DPH is its ability to absorb small, hydrophobic molecules from solution [92, 151]. This was observed in experimental data here, with reductions in the concentration of DPH measured before and after use in both microfluidic devices.

One further issue with both microfluidic devices is the requirement for large sample volumes. The system as tested is an open fluid system and uses a pump. The ideal setup would be a closed fluid system with a pump that could deal with microlitres rather than millilitres. In addition, the use of an HPLC pump was considered to pose a risk to maintaining the vesicle size distribution of samples. However, a comparison of vesicle solution DLS data shown in Figures 5.11, 5.14, F.5 and F.8 indicates that small changes occured in the spread of vesicle diameters after use in microfluidic experiments. This is also shown in the mean vesicle diameter data, provided in Table 5.2.

In contrast with the DLS data, the LD spectroscopic data did indicate that size changes may have occured after use in the microfluidic devices, as indicated by the reduced slope caused by light scattering. Furthermore, this decrease was more prominent with vesicles (with and without DPH) in the deep wide channel than in the deep cross-slot. However, the theory of polarised light scattering is not fully understood, so it is harder to draw con-

Table 5.2: Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 25 samples.

Microfluidic device	Vesicle Solution		Mean / nm	Standard Deviation / nm
Deep wide channel	Without DPH	Before use	340	24
		After use	330	15
	With DPH	Before use	208	6.3
		After use	200	4.4
Deep cross-slot	Without DPH	Before use	330	14
		After use	330	22
	With DPH	Before use	233	6.0
		After use	230	15

clusions on vesicles sizes from LD data than with the data from the established technique of dynamic light scattering.

5.7 Summary and future work

In this chapter, LD experiments have been reported on vesicles samples in pressure-driven, extensional and Couette shear flows. The results indicate that, while it is possible to align vesicles in flow-through microfluidic devices, with a view to obtaining LD spectroscopic data from molecules embedded in the membranes, Couette flow is more optimal than the use of these devices. In particular, pressure driven flows through a wide channel microfluidic device were shown to give an LD signal from DPH molecules embedded in vesicle mebranes, whereas results with extensional flows using a cross-slot device were inconclusive.

Further studies of vesicles in extensional flow for use in LD experiments require optimisation of the cross-slot device used in this project. These include the use of quartz instead of PDMS to avoid both changes in path length at higher flow rates and the removal of DPH molecules from vesicles during flow through the device, and the use of microfluidic pump to allow for smooth flow of fluid through the device, to ensure a balanced and maintained extensional flow.

Chapter 6

Conclusions and future work

The original aim of this thesis was to study the behaviour of molecules in fluid flow, with a view to informing the design of linear dichroism spectroscopy measurements. The scope of the project was split into two main parts: a theoretical component, where the use of bead-spring simulations and the implementation of a particular spring force law were considered, and an experimental component, involving the use of flow-through microfluidic devices to align DNA molecules and vesicles for LD experiments.

6.1 Implementation of the FENE-Fraenkel force law into bead-spring chain simulations

The FENE-Fraenkel spring force law was been implemented in code previously used to study bead-FENE-spring chains in flow. This allowed springs in the bead-spring chain simulations to exhibit behaviour similar to rigid rods as well as flexible springs. To accommodate this spring force law, it was found that the precision level to which variables were being stored by the code needed to be increased.

Initial tests were performed with the code to ascertain if outputs of the spring length for a dumbbell would be close to analytically derived values in zero-shear situations. The results were favourable, with simulated values for the squared length of a spring being no greater than 0.01% away from analytically derived. A small number of simulations with dumbbells in non-zero-shear conditions were also performed to check if the simulated

spring length would respond to changes in the shear rate. The results indicate that spring lengths would increase with increasing shear rate. However, any future work with this code would seek to extend this testing to use larger shear rates to establish more fully the relationship between the length of a dumbbell and the shear rate applied.

Comparisons were then made with the work of Hsieh et al. [48]; this work was the only previously publsihed work known to have studied the FENE-Fraenkel force law in the context of bead-spring simulations. Values of the first normal polymer stress coefficient from simulations of chains consisting of 11 beads and 10 FENE-Fraenkel springs were compared against the data of Hsieh et al. [48]. The differences were no greater than 10%; with the selection of small time-step size, these differences were reduced to the order of 1%. Furthermore, the scale of "noise" present in simulation outputs increased with parameter choices leading to a stiffer spring; this was also observed by Hsieh et al. [48] in their simulations. However the scale of this noise is larger in the simulation results published here. It is postulated that the changes made to the code to enable the use of FENE-Fraenkel springs may be an underlying cause of this. Overall, the code used in simulations for this work does provide good agreement with the work of Hsieh et al. [48], suggesting that the FENE-Fraenkel spring force law has been implemented correctly.

Finally, simulations of the alignment of FENE-Fraenkel spring dumbbells in shear flows show increased alignment with high shear rates, similar to modelling data published by McLachlan et al. [89] which studied the orientation of rods in shear flow. Hydrodynamic interactions were also included and were observed to have no noticable effect on the orientation of dumbbells in shear flow. Since hydrodynamic interactions occur between different parts of a chain, there should not be any effect with dumbbells. Therefore, the code outputs were sensible in this regard.

The work in this thesis demonstrates that our code is able to produce sensible outputs with bead-spring chains using FENE-Fraenkel springs in a limited range of scenarios. Going forward, since the simulations are designed with the study of polymers in mind, any future work with this code would involve simulations of chains containing more than 2 beads in shear and extensional flow. These simulations would be performed with a wider range of strain rates, natural length and extensibility parameter values than those used here. The aim of these simulations would be to study how the alignment parameter changes over

time in these flow regimes. Treating each of the springs in a chain as a chromophore, this would help to inform us of the linear dichroism signal that we can expect when polymer molecules, such as DNA, are being studied in shear and extensional flow.

Attention would need to be paid to how realistic the choice of parameters are, with respect to the behaviour of polymers in reality. In addition, these simulations should also include excluded-volume and hydrodynamic interactions, particularly as longer chains with more beads will increase the prevalence of these interactions. The code, as published here, is able to implement these interactions.

Two further extensions to the code are also proposed. First, an extension on the inclusion of excluded volume interactions would be the inclusion of bending potentials, as discussed by Holleran and Larson [46]; instead of a freely-jointed chain, there would be restrictions on the proximity of springs to each other, reflecting the finite flexibility of polymers in real life. Second, the code here simulates chains in "dilute" conditions, whereby interactions between chains are not considered. The next step would be to consider multiple chains in a simulation and model the intermolecular interactions in addition to those

6.2 Development of an extensional flow microfluidic device for use in LD experiments

The use of microfluidic devices, made with PDMS using standard lithographic techniques, in LD experiments was considered, with two main designs being considered: a "wide channel" device with pressure driven flow and a cross-slot, based on a design published by Haward et al. [44], with extensional flow. These were tested with solutions of calf-thymus DNA. After modifications to increase the channel depth to increase the path length and, by the Beer-Lambert law, the absorbance, each microfluidic device was shown to be able to obtain LD data from DNA solutions. Furthermore, the deep cross-slot device was determined to be able to align a higher proportion of DNA molecules at high flow rates than the deep wide channel device. It is thought that the diminishing ability of the deep wide channel device to align at higher flow rates is due to the thin PDMS wall bulging out with high fluid pressures.

The alignment of DNA molecules using the deep cross-slot device was, however, shown to be poorer than that obtained in the Couette flow cell; the latter being the established means of aligning molecules for LD experiments in our lab [83]. Furthermore, the sample volumes required for the microfluidic devices are two orders of magnitude greater than for Couette flow. Therefore, while the use of flow-through microfluidic devices in LD experiments is possible, the practicality of the devices tested render them inferior to the Couette flow cell.

Linear dichroism experiments were also performed on solutions of soy bean PC lipid vesicle with DPH molecules in vesicle membranes. The motivation for this was to ascertain if the extensional flow in the cross-slot device or the pressure driven flow in the wide channel device could be used to orient vesicles to obtain an LD signal from the DPH molecules, with a view to extending this concept to LD experiments with membrane proteins. The theory on vesicles in fluid flow suggested that an extensional flow would stretch vesicles, thus allowing them to be aligned for the purpose of an LD experiment.

The results indicate that, while it is possible to align vesicles in flow-through microfluidic devices, with a view to obtaining LD spectroscopic data from molecules embedded in the membranes, the use of a Couette flow cell to align the vesicles provided the best results. In particular, pressure driven flows through a wide channel microfluidic device were shown to give an LD signal from DPH molecules embedded in vesicle mebranes, whereas results with extensional flows using a cross-slot device were inconclusive, due to the noise present in the spectra measured.

Further work with these microfluidic devices would require a substantial redesign of the experimental setup. One of the major concerns is the requirement for large sample volumes. The system as tested is an open fluid system and uses a pump. The ideal setup would be a closed fluid system with a pump that could deal with microlitres rather than millilitres. The channel designs would also need to be modified to reduce the sample volumes utilised, whilst allowing absorbance spectra to be measured accurately. The use of quartz rather than PDMS would help with this issue, as quartz is more transparent than PDMS to UV and visible light, and so absorbance and LD spectra could be measured more easily.

It would also avoid the problem of channel walls bulging due to high fluid pressures

within, as observed in some experiments. In addition, PDMS is able to to absorb small, hydrophobic molecules from solution [92, 151]; this is a problem in the context of using molecules such as DPH in samples. The use of quartz would eliminate this issue.

The extensional flow cross-slot device could also be modified in a number of different ways. A major obstacle to the maintenance of a pure extensional flow in the centre of the device is the difficulty of maintaining even rates of fluid flow into the central chamber. The formation and trapping of bubbles in the system can cause fluid flows to be blocked and the flow in the centre to not be extensional. The relatively large channel depths of the device also make it harder to remove bubbles trapped in the device. Furthermore, the use of one pump and a junction to split fluid flow evenly is not optimal.

A remodelled cross-slot device may also require a redesign of the central chamber. The design used here is based on work by Haward et al. [44], who developed the design to maximise the volume of fluid subject to extensional strain. However, Galindo-Rosales et al. [39] have commented that this is based on a 2D flow approximation that does not translate to PDMS microfluidic devices, where end-walls distort the flow field and the device is not able to generate a uniform well-defined extension rate inside the cross-chamber. They have therefore proposed new designs that could be considered for the central chamber.

Future work with the cross-slot microfluidic device could be performed in conjunction with bead-spring chain simulations, to compare predictions of the behaviour of DNA and other polymers in extensional flow with experimental observations. These experiments could be performed with fixed lengths of DNA, obtained, for example, by cutting pre-assembled plasmids in one place. The LD and derived molecular alignment measurements made with in these experiments could then be compared with calculations of predicted alignment parameter values in simulations of bead-spring chains in fluid flow. Finally, if an extensional device is developed that can extend and orient vesicles to obtain a LD spectrum, the data from these experiments could be studied in conjunction with simulations of dumbbells in extensional flow, where elongated vesicles could be modelled as dumbbells.

Bibliography

[1] M. Abkarian, C. Lartigue, and A. Viallat. Tank Treading and Unbinding of Deformable Vesicles in Shear Flow: Determination of the Lift Force. *Phys. Rev. Lett.*, 88:068103, 2002.

[2] D. Abreu, M. Levant, V. Steinberg, and U. Seifert. Fluid vesicles in flow. *Advances in Colloid and Interface Science*, 208(Supplement C):129 – 141, 2014.

[3] R. Adachi, K. Yamaguchi, H. Yagi, K. Sakurai, H. Naiki, and Y. Goto. Flow-induced Alignment of Amyloid Protofilaments Revealed by Linear Dichroism. *Journal of Biological Chemistry*, 282(12):8978 – 8983, 2007.

[4] M. A. Alves. Design of a Cross-Slot Flow Channel for Extensional Viscosity Measurements. *AIP Conference Proceedings*, 1027(1):240 – 242, 2008.

[5] M. Ardhammar, N. Mikati, and B. Norden. Chromophore Orientation in Liposome Membranes Probed with Flow Dichroism. *Journal of the American Chemical Society*, 120(38):9957 – 9958, 1998.

[6] E. D. T. Atkins and M. A. Taylor. Elongational flow studies on DNA in aqueous solution and stress-induced scission of the double helix. *Biopolymers*, 32(8):911 – 923, 1992.

[7] H. P. Babcock, D. E. Smith, J. S. Hur, E. S. G. Shaqfeh, and S. Chu. Relating the Microscopic and Macroscopic Response of a Polymeric Fluid in a Shearing Flow. *Phys. Rev. Lett.*, 85:2018 – 2021, 2000.

[8] C. N. Banwell and E. M. McCash. *Fundamentals of Molecular Spectroscopy*. McGraw-Hill, fourth edition, 1994.

[9] J. M. Berg, J. L. Tymoczko, and L. Stryer. *Biochemistry*. W. H. Freeman and Company, New York, seventh edition, 2012.

[10] I. Berlman. *Handbook of Fluorescence Spectra of Aromatic Molecules*. Academic Press, second edition, 1971.

[11] T. Biben, A. Farutin, and C. Misbah. Three-dimensional vesicles under shear flow: Numerical study of dynamics and phase diagram. *Phys. Rev. E*, 83:031921, 2011.

[12] P. Biller, F. Petruccione, J. Honerkamp, and H. C. Öttinger. Nonlinear dumbbell model for flexible polymers: dynamical phenomena. *Journal of Non-Newtonian Fluid Mechanics*, 22(3):309 – 324, 1987.

[13] R. B. Bird, H. R. Warner, and D. C. Evans. *Kinetic theory and rheology of dumbbell suspensions with Brownian motion*, pages 1 – 90. Springer Berlin Heidelberg, Berlin, Heidelberg, 1971.

[14] R. B. Bird, R. C. Armstrong, and O. Hassager. *Dynamics of Polymeric Liquids*, volume 2. Wiley, 1987.

[15] M. Bloemendal and R. van Grondelle. Linear-dichroism spectroscopy for the study of structural properties of proteins. *Molecular Biology Reports*, 18(1):49 – 69, 1993.

[16] M. Bloom and Mouritsen. *The Evolution of Membranes*, volume 1A of *Handbook of Biological Physics*, chapter 2. Elsevier, 1995.

[17] H. Bruus. *Theoretical Microfluidics*. Oxford University Press, 2007.

[18] B. M. Bulheller, A. Rodger, and J. D. Hirst. Circular and linear dichroism of proteins. *Phys. Chem. Chem. Phys.*, 9:2020 – 2035, 2007.

[19] B. M. Bulheller, A. Rodger, M. R. Hicks, T. R. Dafforn, L. C. Serpell, K. E. Marshall, E. H. C. Bromley, P. J. S. King, K. J. Channon, D. N. Woolfson, and J. D. Hirst. Flow Linear Dichroism of Some Prototypical Proteins. *Journal of the American Chemical Society*, 131(37):13305 — 13314, 2009.

[20] M. Caffrey. Membrane protein crystallization. *Journal of Structural Biology*, 142 (1):108 – 132, 2003.

[21] E. P. Carpenter, K. Beis, A. D. Cameron, and S. Iwata. Overcoming the challenges of membrane protein crystallography. *Current Opinion in Structural Biology*, 18 (5):581 – 586, 2008.

[22] L. F. Cavalieri, B. H. Rosenberg, and M. Rosoff. Flow Dichroism and its Application to the Study of Deoxyribonucleic Acid Structure. *Journal of the American Chemical Society*, 78(20):5235 – 5238, 1956.

[23] M. Couette. Études sur le Frottement des Liquides. *Annales de chimie et de physique*, 6:433 – 510, 1890.

[24] T. R. Dafforn and A. Rodger. Linear dichroism of biomolecules: which way is up? *Current Opinion in Structural Biology*, 14(5):541 – 546, 2004.

[25] G. Danker, T. Biben, T. Podgorski, C. Verdier, and C. Misbah. Dynamics and rheology of a dilute suspension of vesicles: Higher-order theory. *Phys. Rev. E*, 76: 041905, 2007.

[26] J. Deschamps, V. Kantsler, E. Segre, and V. Steinberg. Dynamics of a vesicle in general flow. *Proceedings of the National Academy of Sciences*, 106(28):11444 – 11447, 2009.

[27] J. Deschamps, V. Kantsler, and V. Steinberg. Phase Diagram of Single Vesicle Dynamical States in Shear Flow. *Phys. Rev. Lett.*, 102:118105, 2009.

[28] R. C. DiPrima, P. M. Eagles, and B. S. Ng. The effect of radius ratio on the stability of Couette flow and Taylor vortex flow. *The Physics of Fluids*, 27(10):2403 – 2411, 1984.

[29] P. S. Doyle, E. S. G. Shaqfeh, and A. P. Gast. Dynamic simulation of freely draining flexible polymers in steady linear flows. *Journal of Fluid Mechanics*, 334:251 – 291, 1997.

[30] J. Drenth. *Principles of Protein X-Ray Crystallography*. Springer, third edition, 2007.

[31] D. C. Duffy, J. C. McDonald, O. J. A. Schueller, and G. M. Whitesides. Rapid Prototyping of Microfluidic Systems in Poly(dimethylsiloxane). *Analytical Chemistry*, 70(23):4974 – 4984, 1998.

[32] B. Dupuy and M. Montagu. Spectral Properties of a Fluorescent Probe, all-trans-1,6-Diphenyl-1,3,5-hexatriene. Solvent and Temperature Effects. *Analyst*, 122:783 – 786, 1997.

[33] D. Dyer, S. Shreim, S. Jayadev, V. Lew, E. Botvinick, and M. Khine. Sequential shrink photolithography for plastic microlens arrays. *Applied Physics Letters*, 99 (3):034102, 2011.

[34] A. Farutin, T. Biben, and C. Misbah. Analytical progress in the theory of vesicles under linear flow. *Phys. Rev. E*, 81:061904, 2010.

[35] S. W. Fetsko and P. T. Cummings. Simulation of bead-and-spring chain models for semidilute polymer solutions in shear flow. *International Journal of Thermophysics*, 15(6):1085 – 1091, 1994.

[36] P. J. Flory. The Configuration of Real Polymer Chains. *The Journal of Chemical Physics*, 17(3):303 – 310, 1949.

[37] G. K. Fraenkel. Visco-Elastic Effect in Solutions of Simple Particles. *The Journal of Chemical Physics*, 20(4):642 – 647, 1952.

[38] B. J. Frisken, C. Asman, and P. J. Patty. Studies of Vesicle Extrusion. *Langmuir*, 16(3):928 – 933, 2000.

[39] F. J. Galindo-Rosales, M. S. N. Oliveira, and M. A. Alves. Optimized cross-slot microdevices for homogeneous extension. *RSC Advances*, 4:7799 – 7804, 2014.

[40] E. L. Gilroy, M. R. Hicks, D. J. Smith, and A. Rodger. Viscosity of aqueous DNA solutions determined using dynamic light scattering. *Analyst*, 136:4159 – 4163, 2011.

[41] A. C. Gracetto, V. R. Batistela, W. Caetano, H. P. M. de Oliveira, W. G. Santos, C. C. S. Cavalheiro, and N. Hioka. Unusual 1,6-diphenyl-1,3,5-hexatriene (DPH) spectrophotometric behavior in water/ethanol and water/DMSO mixtures. *Journal of the Brazilian Chemical Society*, 21:1497 – 1502, 2010.

[42] A. Grimes, D. N. Breslauer, M. Long, J. Pegan, L. P. Lee, and M. Khine. Shrinky-Dink microfluidics: rapid generation of deep and rounded patterns. *Lab Chip*, 8: 170 – 172, 2008.

[43] S. J. Haward, T. J. Ober, M. S. N. Oliveira, M. A. Alves, and G. H. McKinley. Extensional rheology and elastic instabilities of a wormlike micellar solution in a microfluidic cross-slot device. *Soft Matter*, 8:536 – 555, 2012.

[44] S. J. Haward, M. S. N. Oliveira, M. A. Alves, and G. H. McKinley. Optimized Cross-Slot Flow Geometry for Microfluidic Extensional Rheometry. *Phys. Rev. Lett.*, 109:128301, 2012.

[45] S. Higashi, M. Kasai, F. Oosawa, and A. Wada. Ultraviolet dichroism of F-actin oriented by flow. *Journal of Molecular Biology*, 7(4):421 – 430, 1963.

[46] S. P. Holleran and R. G. Larson. Using spring repulsions to model entanglement interactions in brownian dynamics simulations of bead-spring chains. *Rheologica Acta*, 47(1):3 – 17, 2008.

[47] J. C. M. Holthuis and A. K. Menon. Lipid landscapes and pipelines in membrane homeostasis. *Nature*, 510(7503):48 – 57, 2014.

[48] C.-C. Hsieh, S. Jain, and R. G. Larson. Brownian dynamics simulations with stiff finitely extensible nonlinear elastic-Fraenkel springs as approximations to rods in bead-rod models. *The Journal of Chemical Physics*, 124(4), 2006.

[49] S. D. Hudson, F. R. Phelan Jr., M. D. Handler, J. T. Cabral, K. B. Migler, and E. J. Amis. Microfluidic analog of the four-roll mill. *Applied Physics Letters*, 85(2):335 – 337, 2004.

[50] R. R. Huilgol and N. Phan-Thien. *Fluid Mechanics of Viscoelasticity: General Principles, Constitutive Modelling, Analytical and Numerical Techniques*, volume 6 of *Rheology Series*. Elsevier, 1997.

[51] I. Hurjui, L.-M. Ivan, and D. O. Dorohoi. Solvent influence on the electronic absorption spectra (EAS) of 1,6-diphenyl-1,3,5-hexatriene (DPH). *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 102:219 – 225, 2013.

[52] Malvern Instruments. *Zetasizer Nano Series User Manual*, 2004.

[53] A. Jain, P. Sunthar, B. Dünweg, and J. Ravi Prakash. Optimization of a Brownian-dynamics algorithm for semidilute polymer solutions. *Phys. Rev. E*, 85(6):066703, 2012.

[54] R. M. Jendrejack, M. D. Graham, and J. J. de Pablo. Hydrodynamic interactions in long chain polymers: Application of the Chebyshev polynomial approximation in stochastic simulations. *The Journal of Chemical Physics*, 113(7):2894 – 2900, 2000.

[55] A. Jesorka and O. Orwar. Liposomes: Technologies and analytical applications. *Annual Review of Analytical Chemistry*, 1(1):801 – 832, 2008.

[56] L. P. Kadanoff. *Statistical Physics: Statics, Dynamics and Renormalization*. World Scientific, 2000.

[57] V. Kantsler and V. Steinberg. Orientation and Dynamics of a Vesicle in Tank-Treading Motion in Shear Flow. *Phys. Rev. Lett.*, 95:258101, 2005.

[58] V. Kantsler and V. Steinberg. Transition to Tumbling and Two Regimes of Tumbling Motion of a Vesicle in Shear Flow. *Phys. Rev. Lett.*, 96:036001, 2006.

[59] V. Kantsler, E. Segre, and V. Steinberg. Vesicle Dynamics in Time-Dependent Elongation Flow: Wrinkling Instability. *Phys. Rev. Lett.*, 99:178102, 2007.

[60] V. Kantsler, E. Segre, and V. Steinberg. Dynamics of interacting vesicles and rheology of vesicle suspension in shear flow. *EPL (Europhysics Letters)*, 82(5):58005, 2008.

[61] B. Kaoui, A. Farutin, and C. Misbah. Vesicles under simple shear flow: Elucidating the role of relevant control parameters. *Phys. Rev. E*, 80:061905, 2009.

[62] R. Kaptein and G. Wagner. NMR studies of membrane proteins. *Journal of Biomolecular NMR*, 61(3):181 – 184, 2015.

[63] M. Kasahara and P. C. Hinkle. Reconstitution and purification of the D-glucose transporter from human erythrocytes. *Journal of Biological Chemistry*, 252(20): 7384 – 7390, 1977.

[64] S. R. Keller and R. Skalak. Motion of a tank-treading ellipsoidal particle in a shear flow. *Journal of Fluid Mechanics*, 120:27 — 47, 1982.

[65] B. Kim, E. T. K. Peterson, and I. Papautsky. Long-term stability of plasma oxidized PDMS surfaces. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 5013 – 5016, 2004.

[66] J. G. Kirkwood and J. Riseman. The Intrinsic Viscosities and Diffusion Constants of Flexible Macromolecules in Solution. *The Journal of Chemical Physics*, 16(6): 565 – 573, 1948.

[67] E. L. Koschmieder. *Benard Cells and Taylor Vortices*. Cambridge University Press, 1993.

[68] H. A. Kramers. Het gedrag van macromoleculen in een stroomende vloeistof. *Physica*, 11(1):1 – 19, 1944.

[69] M. Kröger, A. Alba-Pérez, M. Laso, and H. C. Öttinger. Variance reduced Brownian simulation of a bead-spring chain under steady shear flow considering hydrodynamic interaction effects. *The Journal of Chemical Physics*, 113(11):4767 – 4773, 2000.

[70] B.D. Ladbrooke and D. Chapman. Thermal analysis of lipids, proteins and biological membranes a review and summary of some recent studies. *Chemistry and Physics of Lipids*, 3(4):304 – 356, 1969.

[71] V. V. Lebedev, K. S. Turitsyn, and S. S. Vergeles. Nearly spherical vesicles in an external flow. *New Journal of Physics*, 10(4):043044, 2008.

[72] J. S. Lee, R. Dylla-Spears, N. P. Teclemariam, and S. J. Muller. Microfluidic four-roll mill for all flow types. *Applied Physics Letters*, 90(7):074103, 2007.

[73] S. Lee, Y. Lee, H. M. Lee, J. Y. Lee, D. H. Kim, and S. K. Kim. Rotation of Periphery Methylpyridine of meso-Tetrakis(n-N-methylpyridiniumyl)porphyrin (n = 2, 3, 4) and Its Selective Binding to Native and Synthetic DNAs. *Biophysical Journal*, 83(1):371 – 381, 2002.

[74] B. R. Lentz, Y. Barenholz, and T. E. Thompson. Fluorescence depolarization studies of phase transitions and fluidity in phospholipid bilayers. 1. Single component phosphatidylcholine liposomes. *Biochemistry*, 15(20):4521 – 4528, 1976.

[75] A. M. Lesk. *Introduction to Protein Science*. Oxford University Press, second edition, 2010.

[76] M. Levant and V. Steinberg. Amplification of Thermal Noise by Vesicle Dynamics. *Phys. Rev. Lett.*, 109:268103, 2012.

[77] Avanti Polar Lipids. Avanti Mini-Extruder Equipment, August 2017. URL `https://avantilipids.com/divisions/equipment/`.

[78] S. Liu, B. Ashok, and M. Muthukumar. Brownian dynamics simulations of bead-rod-chain in simple shear flow and elongational flow. *Polymer*, 45(4):1383 – 1389, 2004.

[79] K. Lloyd. Stability of Lipid Vesicles in Flow at High Shear Rates. Master's thesis, The University of Warwick, 2013.

[80] A. Mallock. Determination of the Viscosity of Water. *Proceedings of the Royal Society of London*, 45(273-279):126 – 132, 1888.

[81] A. Mallock. III. Experiments on fluid viscosity. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 187:41 – 56, 1896.

[82] R. Marrington, T. R. Dafforn, D. J. Halsall, and A. Rodger. Micro-Volume Couette Flow Sample Orientation for Absorbance and Fluorescence Linear Dichroism. *Biophysical Journal*, 87(3):2002 – 2012, 2004.

[83] R. Marrington, T. R. Dafforn, D. J. Halsall, J. I. MacDonald, M. Hicks, and A. Rodger. Validation of new microvolume Couette flow linear dichroism cells. *Analyst*, 130:1608 – 1616, 2005.

[84] K. Matsuo, Y. Maki, H. Namatame, M. Taniguchi, and K. Gekko. Conformation of membrane-bound proteins revealed by vacuum-ultraviolet circular-dichroism and linear-dichroism spectroscopy. *Proteins: Structure, Function, and Bioinformatics*, 84(3):349 – 359, 2016.

[85] Y. Matsuoka and B. Norden. Linear dichroism studies of nucleic acids. II. Calculation of reduced dichroism curves of A-and B-form DNA. *Biopolymers*, 21(12): 2433 – 2452, 1982.

[86] Y. Matsuoka and B. Norden. Linear dichroism studies of nucleic acid bases in stretched poly(vinyl alcohol) film. molecular orientation and electronic transition moment directions. *The Journal of Physical Chemistry*, 86(8):1378 – 1386, 1982.

[87] J. C. McDonald and G. M. Whitesides. Poly(dimethylsiloxane) as a Material for

Fabricating Microfluidic Devices. *Accounts of Chemical Research*, 35(7):491 – 499, 2002.

[88] J. C. McDonald, D. C. Duffy, J. R. Anderson, D. T. Chiu, H. Wu, O. J. A. Schueller, and G. M. Whitesides. Fabrication of microfluidic systems in poly(dimethylsiloxane). *Electrophoresis*, 21(1):27 – 40, 2000.

[89] J. R. A. McLachlan, D. J. Smith, N. P. Chmel, and A. Rodger. Calculations of flow-induced orientation distributions for analysis of linear dichroism spectroscopy. *Soft Matter*, 9:4977 – 4984, 2013.

[90] M. Miki and K. Mihashi. Fluorescence and flow dichroism of F-actin-ε-adp; the orientation of the ademine plane relative to the long axis of F-actin. *Biophysical Chemistry*, 6(1):101 – 106, 1976.

[91] E. Moldrheim, M. J. Hannon, I. Meistermann, A. Rodger, and E. Sletten. Interaction between a DNA oligonucleotide and a dinuclear iron(II) supramolecular cylinder; an NMR and molecular dynamics study. *JBIC Journal of Biological Inorganic Chemistry*, 7(7):770 – 780, 2002.

[92] R. Mukhopadhyay. When PDMS isn't the best. *Analytical Chemistry*, 79(9):3248 – 3253, 2007.

[93] R. R. C. New. *Liposomes: a practical approach*. The Practical Approach Series. Oxford University Press, 1990.

[94] B. Nguyen, D. Hamelberg, C. Bailly, P. Colson, J. Stanek, R. Brun, S. Neidle, and W. D. Wilson. Characterization of a Novel DNA Minor-Groove Complex. *Biophysical Journal*, 86(2):1028 – 1041, 2004.

[95] D. Nguyen, D. Taylor, K. Qian, N. Norouzi, J. Rasmussen, S. Botzet, M. Lehmann, K. Halverson, and M. Khine. Better shrinkage than Shrinky-Dinks. *Lab Chip*, 10: 1623 – 1626, 2010.

[96] B. Norden, M. Kubista, and T. Kurucsev. Linear dichroism spectroscopy of nucleic acids. *Quarterly Reviews of Biophysics*, 25(1):51 — 170, 1992.

[97] B. Norden, A. Rodger, and T. Dafforn. *Linear Dichroism and Circular Dichroism*. The Royal Society of Chemistry, 2010.

[98] B. Nordén. Applications of linear Dichroism Spectroscopy. *Applied Spectroscopy Reviews*, 14(2):157 – 248, 1978.

[99] T. Ohsawa, H. Miura, and K. Harada. Evaluation of a New Liposome Preparation Technique, the Freeze-Thawing Method, Using L-Asparaginase as a Model Drug. *Chemical & Pharmaceutical Bulletin*, 33(7):2916 – 2923, 1985.

[100] B. Oksendal. *Stochastic Differential Equations*. Springer, 2003.

[101] K. Osaki, T. Inoue, and T. Isomura. Stress overshoot of polymer solutions at high rates of shear. *Journal of Polymer Science Part B: Polymer Physics*, 38(14):1917 – 1925, 2000.

[102] H. C. Öttinger. *Stochastic Processes in Polymeric Fluids — Tools and Examples for Developing Simulation Algorithms*. Springer-Verlag, 1996.

[103] J. P. Overington, B. Al-Lazikani, and A. L. Hopkins. How many drug targets are there? *Nat Rev Drug Discov*, 5(12):993 – 996, 2006.

[104] R. A. Parente and B. R. Lentz. Advantages and limitations of 1-palmitoyl-2-[2-[4-(6-phenyl-trans-1,3,5-hexatrienyl)phenyl]ethylcarbonyl]-3-sn-phosphatidylcholine as a fluorescent membrane probe. *Biochemistry*, 24(22): 6178 – 6185, 1985.

[105] K. K. Patel, E. A. Plummer, M. Darwish, A. Rodger, and M. J. Hannon. Aryl substituted ruthenium bis-terpyridine complexes: intercalation and groove binding with DNA. *Journal of Inorganic Biochemistry*, 91(1):220 – 229, 2002. Contributions from the 10th Intenational Conference on Bioinorganic Chemistry.

[106] T. T. Perkins, D. E. Smith, and S. Chu. Single Polymer Dynamics in an Elongational Flow. *Science*, 276(5321):2016–2021, 1997.

[107] D. Petera and M. Muthukumar. Brownian dynamics simulation of bead–rod chains under shear with hydrodynamic interaction. *The Journal of Chemical Physics*, 111 (16):7614 – 7623, 1999.

[108] T. T. Pham, P. Sunthar, and J. Ravi Prakash. An alternative to the bead-rod model: Bead-spring chains with successive fine graining. *Journal of Non-Newtonian Fluid*

Mechanics, 149(1–3):9–19, 2008. International Workshop on Mesoscale and Multiscale Description of Complex Fluids.

[109] A. Pommella, S. Caserta, and S. Guido. Dynamic flow behaviour of surfactant vesicles under shear flow: role of a multilamellar microstructure. *Soft Matter*, 9: 7545 – 7552, 2013.

[110] C. Pozrikidis. *Introduction to theoretical and computational fluid dynamics*. Oxford University Press, 1997.

[111] R. Prabhakar and J. Ravi Prakash. Viscometric functions for Hookean dumbbells with excluded volume and hydrodynamic interactions. *Journal of Rheology*, 46(5): 1191 – 1220, 2002.

[112] R. Prabhakar and J. Ravi Prakash. Multiplicative separation of the influences of excluded volume, hydrodynamic interactions and finite extensibility on the rheological properties of dilute polymer solutions. *Journal of Non-Newtonian Fluid Mechanics*, 116(2–3):163 – 182, 2004.

[113] R. Prabhakar, J. Ravi Prakash, and T. Sridhar. A successive fine-graining scheme for predicting the rheological properties of dilute polymer solutions. *Journal of Rheology*, 48(6):1251 – 1278, 2004.

[114] J. Ravi Prakash and H. C. Öttinger. Viscometric Functions for a Dilute Solution of Polymers in a Good Solvent. *Macromolecules*, 32(6):2028 – 2043, 1999.

[115] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[116] J. Rajendra, A. Damianoglou, M. Hicks, P. Booth, P. M. Rodger, and A. Rodger. Quantitation of protein orientation in flow-oriented unilamellar liposomes by linear dichroism. *Chemical Physics*, 326(1):210 – 220, 2006. The Molecules and Methods of Chemical, Biochemical and Nanoscale Electron Transfer, Part 2.

[117] P. Ranganathan and J.R. Prakash. Rheological behavior of FENE dumbbells with excluded volume interactions in simple shear and uniaxial extensional flows. In

P. A. Gostomski and K. R. Morison, editors, *Proceedings of the 9th APCChe Congress and CHEMECA 2002*, pages 428 – 446. University of Cantebury, 2002.

[118] M. Rittman, E. Gilroy, H. Koohy, A. Rodger, and A. Richards. Is DNA a worm-like chain in Couette flow? In search of persistence length, a critical review. *Science Progress*, 92(2):163 – 204, 2009.

[119] V. Rizzo and J. Schellman. Flow dichroism of t7 dna as a function of salt concentration. *Biopolymers*, 20(10):2143 – 2163, 1981.

[120] A. Rodger, A. Parkinson, and S. Best. Molecular Features of CoIII Tetra- and Pentammines Affect Their Influence on DNA Structure. *European Journal of Inorganic Chemistry*, 2001(9):2311 – 2316, 2001.

[121] A. Rodger, J. Rajendra, R. Marrington, M. Ardhammar, B. Norden, J. D. Hirst, A. T. B. Gilbert, T. R. Dafforn, D. J. Halsall, C. A. Woolhead, C. Robinson, T. J. T. Pinheiro, J. Kazlauskaite, M. Seymour, N. Perez, and M. J. Hannon. Flow oriented linear dichroism to probe protein orientation in membrane environments. *Phys. Chem. Chem. Phys.*, 4:4051 – 4057, 2002.

[122] A. Rodger, R. Marrington, M. A. Geeves, M. Hicks, L. de Alwis, D. J. Halsall, and T. R. Dafforn. Looking at long molecules in solution: what happens when they are subjected to Couette flow? *Phys. Chem. Chem. Phys.*, 8:3161 – 3171, 2006.

[123] J. Rotne and S. Prager. Variational Treatment of Hydrodynamic Interaction in Polymers. *The Journal of Chemical Physics*, 50(11):4831 – 4837, 1969.

[124] P. E. Rouse. A Theory of the Linear Viscoelastic Properties of Dilute Solutions of Coiling Polymers. *The Journal of Chemical Physics*, 21(7):1272 – 1280, 1953.

[125] D. Russell and J. Sambrook. *Molecular Cloning: A Laboratory Manual*, volume 3. Cold Spring Harbor, third edition, 2001.

[126] C. R. Sanders and F. Sönnichsen. Solution NMR of membrane proteins: practice and challenges. *Magnetic Resonance in Chemistry*, 44(S1):S24 – S40, 2006.

[127] N. Sasaki, I. Hayakawa, K. Hikichi, and E. D. T. Atkins. Deformation of λ-phage DNA molecules in an elongational flow field. *Journal of Applied Polymer Science*, 59(9):1389 – 1394, 1996.

[128] N. Sasaki, Y. Maki, and M. Nakata. Elongation flow studies of DNA as a function of temperature. *Journal of Applied Polymer Science*, 83(6):1357 – 1365, 2002.

[129] W. Schärtl. *Light Scattering from Polymer Solutions and Nanoparticle Dispersions*. Springer Laboratory. Springer, 2007.

[130] D. Schneider and D. M. Engelman. GALLEX, a Measurement of Heterologous Association of Transmembrane Helices in a Biological Membrane. *Journal of Biological Chemistry*, 278(5):3105 – 3111, 2003.

[131] B. Schoepp, E. Chabaud, C. Breyton, A. Verméglio, and J. Popot. On the Spatial Organization of Hemes and Chlorophyll in cytochrome b_6f: A LINEAR AND CIRCULAR DICHROISM STUDY. *Journal of Biological Chemistry*, 275(8):5275 – 5283, 2000.

[132] F. Schultz-Grunow. Zur Stabilität der Couette-Strömung. *Z. angew. Math. Mech.*, 39:101 – 110, 1959.

[133] U. Seifert. *Giant Vesicles*, volume Sixth of *Perspectives in Supramolecular Chemistry*, chapter 7, pages 71 – 99. John Wiley & Sons, Ltd., 2000.

[134] J. H. Seinfeld and S. N. Pandis. *Atmospheric Chemistry and Physics*. John Wiley and Sons, second edition, 2006.

[135] E. S. G. Shaqfeh. The dynamics of single-molecule DNA in flow. *Journal of Non-Newtonian Fluid Mechanics*, 130(1):1 – 28, 2005.

[136] M. Shinitzky and Y. Barenholz. Dynamics of the Hydrocarbon Layer in Liposomes of Lecithin and Sphingomyelin Containing Dicetylphosphate. *Journal of Biological Chemistry*, 249(8):2652 – 2657, 1974.

[137] M. P. Short and S. Yip. Materials aging at the mesoscale: Kinetics of thermal, stress, radiation activations. *Current Opinion in Solid State and Materials Science*, 19(4):245 – 252, 2015.

[138] T. Simonson and M. Kubista. DNA orientation in shear flow. *Biopolymers*, 33(8): 1225 – 1235, 1993.

[139] M. Somasi, B. Khomami, N. J. Woo, J. S. Hur, and E. S. G. Shaqfeh. Brownian dynamics simulations of bead-rod and bead-spring chains: numerical algorithms

and coarse-graining issues. *Journal of Non-Newtonian Fluid Mechanics*, 108(1–3): 227 – 255, 2002. Numerical Methods Workshop S.I.

[140] F. J. Stadler, M. Rajan, U. S. Agarwal, C.-Y. Liu, K. E. George, P. J. Lemstra, and C. Bailly. Rheological characterization in shear of a model dumbbell polymer concentrated solution. *Rheologica Acta*, 50(5):491, June 2011.

[141] G. G. Stokes. *Mathematical and Physical Papers*. Cambridge University Press, 1845.

[142] S. R. Strand, S. Kim, and S. J. Karrila. Computation of rheological properties of suspensions of rigid rods: Stress growth after inception of steady shear flow. *Journal of Non-Newtonian Fluid Mechanics*, 24:311 – 329, 1987.

[143] P. Sunthar and J. Ravi Prakash. Parameter-Free Prediction of DNA Conformations in Elongational Flow by Successive Fine Graining. *Macromolecules*, 38(2):617 – 640, 2005.

[144] P. Tabeling. *Introduction to Microfluidics*. Oxford University Press, 2010.

[145] S. H. Tan, N. Nguyen, Y. C. Chua, and T. G. Kang. Oxygen plasma treatment for reducing hydrophobicity of a sealed polydimethylsiloxane microchannel. *Biomicrofluidics*, 4(3):032204, 2010.

[146] C. Tannous. Orientation control of rodlike objects by flow. *ArXiv e-prints*, 2009.

[147] M. Tanyeri, E. M. Johnson-Chavarria, and C. M. Schroeder. Hydrodynamic trap for single particles and cells. *Applied Physics Letters*, 96(22):224101, 2010.

[148] M. Tanyeri, M. Ranka, N. Sittipolkul, and C. M. Schroeder. A microfluidic-based hydrodynamic trap: design and implementation. *Lab Chip*, 11:1786 – 1794, 2011.

[149] G. I. Taylor. Stability of a Viscous Liquid Contained between Two Rotating Cylinders. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 223(605-615):289 – 343, 1923.

[150] G. I. Taylor. The Formation of Emulsions in Definable Fields of Flow. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 146(858):501 – 523, 1934.

[151] M. W. Toepke and D. J. Beebe. PDMS absorption of small molecules and consequences in microfluidic applications. *Lab Chip*, 6:1484 – 1486, 2006.

[152] M. Traïkia, D. E. Warschawski, M. Recouvreur, J. Cartaud, and P. F. Devaux. Formation of unilamellar vesicles by repetitive freeze-thaw cycles: characterization by electron microscopy and 31p-nuclear magnetic resonance. *European Biophysics Journal*, 29(3):184 – 195, 2000.

[153] K. S. Turitsyn and S. S. Vergeles. Wrinkling of vesicles during transient dynamics in elongational flow. *Phys. Rev. Lett.*, 100:028103, 2008.

[154] P. T. Underhill and P. S. Doyle. On the coarse-graining of polymers into bead-spring chains. *Journal of Non-Newtonian Fluid Mechanics*, 122(1—3):3 – 31, 2004. {XIIIth} International Workshop on Numerical Methods for Non-Newtonian Flows.

[155] H. van Amerongen and R. van Grondelle. Orientation of the bases of single-stranded DNA and polynucleotides in complexes formed with the gene 32 protein of bacteriophage T4: A linear dichroism study. *Journal of Molecular Biology*, 209 (3):433 – 445, 1989.

[156] L. Dalla Via, O. Gia, S. M. Magno, A. Da Settimo, G. Primofiore, F. Da Settimo, F. Simorini, and A. M. Marini. Dialkylaminoalkylindolonaphthyridines as potential antitumour agents: synthesis, cytotoxicity and DNA binding properties. *European Journal of Medicinal Chemistry*, 37(6):475 – 486, 2002.

[157] A. Wada. Chain regularity and flow dichroism of deoxyribonucleic acids in solution. *Biopolymers*, 2(4):361 – 380, 1964.

[158] A. Wada. Dichroic Spectra of Biopolymers Oriented by Flow. *Applied Spectroscopy Reviews*, 6(1):1 – 30, 1972.

[159] K. Wakabayashi, N. Sasaki, and K. Hikichi. Elongational flow studies of the conformation of DNA molecules in the globular state. *Journal of Applied Polymer Science*, 76(8):1351 – 1358, 2000.

[160] B. A. Wallace and R. W. Janes. *Modern Techniques for Circular Dichroism and Synchrotron Radiation Circular Dichroism Spectroscopy*. IOS Press, June 2009.

[161] H. R. Warner. Kinetic Theory and Rheology of Dilute Suspensions of Finitely Extendible Dumbbells. *Industrial & Engineering Chemistry Fundamentals*, 11(3): 379 – 387, 1972.

[162] M. C. Wendl. General solution for the Couette flow profile. *Phys. Rev. E*, 60:6192 – 6194, 1999.

[163] G. Wilemski and G. Tanaka. Efforts toward an exact Kirkwood-Riseman theory of the intrinsic viscosity. *Macromolecules*, 14(5):1531 – 1538, 1981.

[164] R. W. Wilson and J. A. Schellman. The flow linear dichroism of DNA: Comparison with the bead-spring theory. *Biopolymers*, 17(5):1235 – 1248, 1978.

[165] H. Yamakawa. Transport Properties of Polymer Chains in Dilute Solution: Hydrodynamic Interaction. *The Journal of Chemical Physics*, 53(1):436 – 443, 1970.

[166] A. Yazdani and P. Bagchi. Three-dimensional numerical simulation of vesicle dynamics using a front-tracking method. *Phys. Rev. E*, 85:056308, 2012.

[167] H. Zhao and E. S. G. Shaqfeh. The dynamics of a vesicle in simple shear flow. *Journal of Fluid Mechanics*, 674:578 — 604, 2011.

[168] H. Zhao and E. S. G. Shaqfeh. The shape stability of a lipid vesicle in a uniaxial extensional flow. *Journal of Fluid Mechanics*, 719:345–361, 2013.

[169] B. H. Zimm. Dynamics of polymer molecules in dilute solution: Viscoelasticity, flow birefringence and dielectric loss. *The Journal of Chemical Physics*, 24(2):269 – 278, 1956.

Appendix A

Derivation of Equation (2.4.11)

Recall the definition of a *Beta function*.

Definition (Beta Functions)**.** A *Beta function* is a function $\mathrm{B}(x, y)$ on two complex numbers, $x, y \in \mathbb{C}$ with $\mathrm{Re}(x), \mathrm{Re}(y) > 0$, given by the following integral:

$$\mathrm{B}(x, y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,\mathrm{d}t.$$

The following result is derived here.

Lemma. For any $x, y \in \mathbb{C}$ with $\mathrm{Re}(x), \mathrm{Re}(y) > 0$,

$$\mathrm{B}(x+1, y) = \mathrm{B}(x, y)\frac{x}{x+y}.$$

To show this, it will be necessary to use *Gamma functions*.

Definition (Gamma Functions)**.** A *Gamma function* $\Gamma(x)$ is a function of a single complex number, $x \in \mathbb{C}$ with $\mathrm{Re}(x) > 0$, and are defined by the following integral:

$$\Gamma(x) := \int_0^\infty e^{-u}u^{x-1}\,\mathrm{d}u. \tag{A.0.1}$$

Using integration by parts,

$$\begin{aligned}\Gamma(x+1) &= \int_0^\infty e^{-u}u^x \,\mathrm{d}u = -\int_0^\infty u^x \left(\frac{\mathrm{d}}{\mathrm{d}u}e^{-u}\right)\mathrm{d}u \\ &= -\left[u^x e^{-u}\right]_0^\infty + \int_0^\infty e^{-u}\left(\frac{\mathrm{d}}{\mathrm{d}u}u^x\right)\mathrm{d}u \\ &= -0 + \int_0^\infty e^{-u}xu^{x-1}\,\mathrm{d}u = x\int_0^\infty e^{-u}xu^{x-1}\,\mathrm{d}u = x\times\Gamma(x)\end{aligned}$$

Furthermore,

$$\begin{aligned}\Gamma(x)\Gamma(y) &= \int_0^\infty e^{-u}u^{x-1}\,\mathrm{d}u\int_0^\infty e^{-v}v^{y-1}\,\mathrm{d}v \\ &= \int_0^\infty\int_0^\infty e^{-(u+v)}u^{x-1}v^{y-1}\,\mathrm{d}u\,\mathrm{d}v.\end{aligned}$$

Now consider the change of variables $u = st$ and $v = s(1-t)$. This gives $u+v = s$, $s \in (0,\infty)$ and $t \in (0,1)$. In addition, the Jacobian for the change is:

$$\frac{\partial(u,v)}{\partial(s,t)} = \begin{vmatrix} t & s \\ (1-t) & -s \end{vmatrix} = -st - s(1-t) = -s.$$

Since $s > 0$, this gives $\left|\frac{\partial(u,v)}{\partial(s,t)}\right| = s$. Therefore,

$$\begin{aligned}\Gamma(x)\Gamma(y) &= \int_0^\infty\int_0^\infty e^{-(u+v)}u^{x-1}v^{y-1}\,\mathrm{d}u\,\mathrm{d}v \\ &= \int_0^1\int_0^\infty e^{-s}(st)^{x-1}(s(1-t))^{y-1}s\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_0^1\int_0^\infty e^{-s}s^{x-1}t^{x-1}s^{y-1}(1-t)^{y-1}s\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_0^1\int_0^\infty e^{-s}s^{x+y-1}t^{x-1}(1-t)^{y-1}\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_0^\infty e^{-s}s^{x+y-1}\,\mathrm{d}s\int_0^1 t^{x-1}(1-t)^{y-1}\,\mathrm{d}t \\ &= \Gamma(x+y)\mathrm{B}(x,y).\end{aligned}$$

Hence,

$$\mathrm{B}(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \tag{A.0.2}$$

Using this and $\Gamma(x+1) = x \times \Gamma(x)$, it can be shown that

$$\mathrm{B}(x+1, y) = \frac{\Gamma(x+1)\Gamma(y)}{\Gamma(x+y+1)} = \frac{x\,\Gamma(x)\Gamma(y)}{(x+y)\,\Gamma(x+y)} = \mathrm{B}(x, y)\frac{x}{x+y}.$$

Appendix B

Fortran code documentation

NOTE: The original code has had white space inserted so that, where possible, changes made (as detailed in **??**) do not affect the position of existing, unmodified lines of code.

B.1 sensemble.f90

This code file contains the main instructions for running the simulation of bead-spring chains in flow and obtaining data for various properties of the chain.

- Lines 1 – 75 contain preamble, including variables to be called, their types and modules to be used (as defined in other code files).
- Line 77 – 103 take the input values for certain parameter from another file, `inputc.dat`. These variables are described in Table B.1.
- Lines 105 – 122 preallocates memory for storing simulation data.
- Lines 129 – 140 determines the simulation time for a chain to equilibrate before the fluid flow is switched on. These are based on formulae derived by Rouse [124] and Zimm.
- Lines 142 – 146 sets the simulation time for a chain when flow is switched on depending on whether or not the shear rate is set to zero.
- Lines 148 – 171 sets the Fortran formatting rules for simulation data outputs.

Table B.1: Input variables in lines 77 to 103 of sensemble.f90.

Code	Expression
`SType`	Spring Type (see line 25 of `modules.f90`)
`NBeads`	Number of beads N
`hstar`	Hydrodynamic interaction parameter h
`zstar`	Exclusion volume force parameter z
`dstar`	Exclusion volume force parameter d
`sqrtb`	FENE extensibility parameter $\sqrt{b}$ (Also used as the FENE-Fraenkel extensibility parameter δ)
`Gdots`	Shear/Extensional strain rate
`emax`	Maximum strain units for the simulation (`emax` = `gdot` $\times$ total time)
`Nsamples`	Number of time points to be sampled over the course of the simulation
`ndelts`	Number of different time-step (Δt) sizes to be used
`deltseqvals`	Time-step Δt for the equilibration part of the simulation
`deltsvals`	Time-step Δt for the flow part of the simulation
`nthisvals`	Number of chains to be simulated on one processor
`ntrajvals`	Number of chains to be simulated in total for whole simulation
`tol`	Tolerance for the absolute error to end the iterations (explained in Subsection 2.7.4)

- Lines 185 – 201 sets the labels for the chain properties that will be measured.
- Lines 204 – 224 prepares an output file `result.dat` which will contain data on the chain properties at the end of the simulation, including the square of the end-to-end chain vector <Rˆ2> and the radius of gyration of the chains <Rgˆ2>, all averaged across all chain simulations.
- Lines 229 – 238 generate a seed, based on the current time, for the random number generators in the program.
- Lines 242 – 555 contain the main simulation code. The loop `timesteps` is a loop over the different sizes of time-steps given in the input file. If there is only one time-step size chosen, then this loop only runs once.
- Lines 248 – 250 select the time-step sizes for the equilibration and flow stages of the simulation and the value for `tol`.
- Lines 252 – 259 calculates the number of time-points at which chain properties will be measured, based on `NSamples`.

- Lines 261 – 290 lets the code search for data from simulations carried out by a different processor in the same run. This feature is provided so that the total number of chains being simulated can be split across different processors using the `nthisvals`, the number of chains to be simulated on on processor, and `ntrajvals`, the total number of chains to be simulated.
- Lines 295 – 342 contains the do loop `trajectories` for running the simulation on one bead-spring chain.
- Line 304 calls the `Initial_position` subroutine from the `utils.f90` file to generate the initial positions for the beads (and springs) in the chain.
- Lines 316 – 320 switches the flow off (equilibrium `EQ` mode) and calls the subroutine `Time_Integrate_Chain` from the `gsipc.f90` file to simulate the movement of the chain over time.
- Lines 324 – 329 calls the `Time_Integrate_Chain` again only if the excluded volume potential is switched on. This is a further equilbration stage to accommodate the extra interactions resulting from excluded volume interactions.
- Lines 332 – 336 switches the shear flow on (`SH` mode — see the `flowvars` module in the `module.f90` file) and calls the `Time_Integrate_Chain` subroutine to run the main simulation.
- Lines 339 – 340 save the data from this chain's simulation.
- Lines 345 – 559 deal with the saving of data from different processors, the calculation of various average chain properties using subroutines from the `properties.f90` file and sends data outputs to:
  - `result.dat` (generated earlier), containing data on chain properties (averaged across all chains) at the end of the simulation.
  - `output.dat`, containing data on chain properties (averaged across all chains) at various time points during the simulation.
- Lines 569 – 601 contains formatting details for various data outputs.

B.2 modules.f90

This file contains code that allows the functions and subroutines from different files to communicate with one another. It also includes specific fixed parameters for the whole program (lines 13 – 27 & 45).

B.3 utils.f90

This code contains utilities that used by the main simulation.

- Subroutine `Initial_position` in lines 14 – 96 generates the initial positions of beads and springs in a chain.
  - Line 36 calls `GaussRand`, which generates a vector of random numbers sampled from a standard normal distribution. This vector, `R`, will eventually contain the positions of all the beads in the chain.
  - Line 40 sets the position of the first bead to the origin.
  - Line 47 recalculates FENE parameter b from the input `L0s` = $\sqrt{b}$.
  - Lines 56 – 95 form a `do` loop over all beads in the chain except the first.
  - Lines 61 – 83 adjust bead positions by calculating suitable values for the length based upon the spring force law being implemented (except for the Hookean force law).
  - Lines 63 & 64 generate a random number from the standard uniform distribution `rv` and then use the `eqpr` function (see below) to change the value of `rv` to a random value for the distance between two connected beads based on the distribution function ϕ (see Subsection 2.4.2).
  - Line 76 generates a separate random number from the standard uniform distribution. This is then used in a rejection sampling algorithm for the value of `rv` (line 80).
  - If the number is accepted, lines 82-83 take the direction of the vector stored in `R` and change the length to match `rv`. The position of the bead being con-

sidered is then altered by adding the bead-to-bead vector `b2b` with length `rv` to the position vector of the previous bead.

- Subroutine `GaussRand` in lines 98 – 132 generates a vector of random variables from a standard Gaussian distribution using a uniform random number generator (`ran_1` in line 119). Details on the mathematics of this generator can be found in Öttinger [102, pp 309–310].
- Subroutine `ran_1` in lines 136 – 177 generates a vector of random variables from a standard uniform distribution. It is based on code detailed in [115, §7.1, p271].
- Subroutine `chbyshb` in lines 180 – 218 calculates the Chebyshev coefficients for the Chebyshev approximation to the square root function. This subroutine is used in the `gsipc.f90` file for calculating **B** by taking the square root of **D** (see Subsection 2.7.1). Further details of this method can be found in [54, 69, 112].
- Subroutine `maxminev_fi` in lines 224 – 261 is used to approximate the maximum and minimum eigenvalues of a matrix. This is required in `gsipc.f90` (line 662) when calculating the square root of **D** to get **B** and $\Delta\mathbf{S}$ using the Chebyshev approximation of the square root function.
- Subroutine `polish_poly_root` in lines 266 – 312 uses the Newton-Raphson algorithm to polish estimates for the roots of polynomials.
- Subroutine `numint` in lines 315 – 359 provides the ability to perform numerical integration and is used in calculating chain properties (lines 481 – 484 of `sensemble.f90`).
- Subroutine `meanerr` in lines 361 – 381 calculates the mean and standard deviation of data given in a vector.
- The `normeqpr` function in lines 383 – 457 calculates $\psi_{eq}(\mathbf{Q})$ in Equation (2.4.4). The integral in the denominator is calculated using a Gaussian quadrature formula, with the 21 quadrature points used listed in lines 409 – 429. The limits of integration are the minimum and maximum lengths of a spring (*e.g.* in the FENE case, 0 and $\sqrt{b}$ respectively).
- The `expphi` function in lines 468 – 487 calculates the value of $\exp(-\phi(\mathbf{Q}))$ as found in Equation (2.4.4), depending on the type of spring being used.

- The `lam1_th` function in lines 490 – 507, used in Line 125 of `sensemble.f90`, calculates the relaxation time of a bead-spring chain according to Zimm theory. The construction of the formula can be found in [50, §34.11, p214].
- Subroutine `get_kappa` in lines 509 – 564 selects the correct form of κ based on the flow type required.
- Lines 567 – 628 deal with the handling of files containing data to be used or written into by the program. The subroutines here are especially important when several different processors are being used to run the program, as explained in Appendix B.1.

B.4 gsipc.f90

This file contains a number of subroutines within modules called by the code in the `sensemble.f90` file, along with the main subroutine of the whole simulation, `Time_Integrate_Chain`.

- Lines 19 – 103 comprise the `csputls` module. This module contains subroutines that extract information about the current state of a bead-spring chain from the variable holding the position vectors of the beads in the chain, `R`.
  - Subroutine `b2bvector_sym_up` (lines 26 – 52) calculates the connector vectors of the springs between beads.
  - Subroutine `modr_sym_up` (lines 54 – 84) calculates the lengths of the spring connector vectors using the data output that comes from the `b2bvector_sym_up` subroutine.
  - Subroutine `tensor_prod_of_vec` (lines 87 – 102) calculates the tensor (or outer) product of a vector, `vec` with itself and multiplied by a scalar, `alpha`.
- Lines 106 – 442 comprise the `BSpModel` module. This contains functions and subroutines that are related to the spring and excluded volume interaction forces.
  - Subroutine `force_sans_hookean` in lines 122 – 153 compute the spring force for a spring connector vector. The expressions for `ff` correspond to

a function f given by the following relationship to $\mathbf{F}^c$.

$$\mathbf{F}^c_\nu(\mathbf{Q}_\nu) = \mathbf{Q}_\nu f\left(\frac{Q}{\sqrt{b}}\right)$$

This is the form used in equation 7 of Pham et al. [108], where $\mathtt{r} = Q/\sqrt{b}$.

- Subroutine `Spring_force` in lines 155 – 192 calculates the forces exerted on a bead by the springs that are connected to it, *i.e.* $\mathbf{F}^S_\nu = \mathbf{F}^c_\nu - \mathbf{F}^c_{\nu-1}$ as given in Subsection 2.7.1.
- Subroutine `solve_implicit_r` in lines 195 – 282 finds solutions to the cubic polynomial as described in Subsection 2.7.5. Rather than finding explicit roots to the cubic, it makes an initial guess to a solution (for example, in the FENE case, line 227) and then uses the `polish_poly_root` subroutine from `utils.f90` to find the root. Using the calculated new length of the spring, it then calculates the new spring tension force $\mathbf{F}^c$ in line 268 by calling the `force_sans_hookean` subroutine.
- Subroutine `Excluded_volume_force` in lines 404 – 441 calculates the foces due to excluded volume interactions between beads, as described in Section 2.5.

- Lines 445 – 1026 house the `Time_Integrate_Chain` subroutine code. This subroutine updates the bead configuration in response to the flow, hydrodynamic interactions, excluded volume forces and spring forces.
  - Lines 450 – 455 describe all the modules to be called in the subroutine.
  - Lines 457 – 548 prepare all variables to be used in the subroutine.
  - Lines 550 – 609 set initial values for certain parameters and variables, including a call to the `chbyshb` subroutine in line 603 to prepare Chebyshev coefficients in a variable `cheba`.
  - Lines 619 – 630 find the centre-of-mass of the bead-spring chain and reset the co-ordinates of all beads so that the chain's centre-of-mass is at the origin.
  - Lines 632 – 635 calculate the bead-to-bead vectors (*i.e.* the spring connector

vectors) and their lengths using two subroutines from the `csputls` module.

- Lines 638 – 641 calculate the forces on the beads due to the springs and excluded volume potentials using two subroutines from the `BSpModel` module.
- Lines 646 – 647 adds the two forces together for each bead.
- Lines 649 – 675 calculate the Rotne-Prager-Yamakawa tensor as mentioned in Subsection 2.7.1 and described in detail in [112, §2].
- Lines 677 – 682 monitor the movement of the chain over time by checking the position of the chain's centre-of-mass.
- Lines 689 – 711 call subroutines from the `properties.f90` file to measure various chain properties (see documentation for `sensemble.f90` and `properties.f90`).
- Lines 716 – 812 generate stochastic matrices **D**, **B** (using the Chebyshev polynomical approximation to the square root function), and

$$\texttt{DelS} = \Delta\mathbf{S} = \frac{1}{\sqrt{2}}\mathbf{B}_n \cdot \Delta\mathbf{W}_n.$$

  These matrices are described in Subsection 2.7.1. The Wiener process is represented by lines 720 – 724.
- Lines 816 – 839 construct the first guess to the new bead configuration, $\widetilde{\mathbf{R}}$ as in Equation (2.7.9) — this is saved as `R_pred`. Lines 821 – 829 calculate the last two terms inside the brackets of Equation (2.7.9), where the condition of `Hstar` = 0 (*i.e.* no hydrodynamics) means that **D** is the unit tensor. Lines 832 – 834 add the $\mathbf{K} \cdot \mathbf{R}$ term, and finally lines 836 – 837 add the remaining terms in Equation (2.7.9).
- Lines 843 – 872 construct $\widetilde{\Upsilon}_{n+1}$ as in Equation (2.7.14). Line 848 starts with:

$$\mathbf{R}_n + \left[\frac{1}{2}\mathbf{K} \cdot \mathbf{R}_n + \frac{1}{8}\mathbf{D}_n \cdot \mathbf{F}^S(\mathbf{R}_n) + \frac{1}{8}\mathbf{F}^E(\mathbf{R}_n)\right]\Delta t + \frac{1}{\sqrt{2}}\mathbf{B}_n \cdot \Delta\mathbf{W}_n$$

  Lines 850 – 866 calculate the $\mathbf{F}^E(\widetilde{\mathbf{R}}_{n+1})$ term for the excluded volume forces based on the predicted bead configuration, and line 872 adds half of this

and $\frac{1}{2}\mathbf{K} \cdot \widetilde{\mathbf{R}}_{n+1}$ to `Ups_pred` to get the final expression for $\widetilde{\Upsilon}_{n+1}$ as in Equation (2.7.14).

- Lines 876 – 896 apply the $\mathcal{D}_\nu$ operator, for all beads $\nu \in \{1, \ldots, N\}$ to `Ups_pred`= $\widetilde{\Upsilon}_{n+1}$ and `Diffusion_sup`= $\mathbf{D}_n$.
- Lines 909 – 1004 (along with some preamble in lines 899 – 906) form the iteration loop as described at the end of Subsection 2.7.4, with lines 775 – 827 describing the process for each spring in the bead-spring chain.
- Line 919 calculates the spring connector force $\mathbf{F}^c$ using the spring forces calculated earlier in line 640 using bead configuration $\mathbf{R}_n$. Lines 926 – 934 calculate the $\frac{1}{8}\mathcal{D}_\nu \left[\mathbf{D}_n \cdot \mathbf{F}^{S,(j-1)}(\mathbf{R}_{n+1})\right] \Delta t$ term in Equation (2.7.17), but use $\mathbf{R}_n$ and not the unknown $\mathbf{R}_{n+1}$. The remaining terms are added in lines 939 – 940 to get the full expression for $\mathbf{\Gamma}^{(j-1)}_{\nu,n+1}$ as in Equation (2.7.17).
- Line 942 calculates the magnitude of $\mathbf{\Gamma}^{(j-1)}_{\nu,n+1}$ before calling the `solve_implicit_r` subroutine (line 952) to solve the cubic described in Equation (2.7.21). Lines 976 – 978 use `gama_mu` to hold the new spring connector vector, noting that $\mathbf{\Gamma}$ is a scalar multiple of $\mathbf{Q}$ as explained in Subsection 2.7.5. Lines 980 – 989 update the existing bead configuration with the new spring length position and spring forces.
- Lines 993 – 1003 provide the conditions to be satisfied for the iterative loop to end — either the change in the calculated bead configurations becomes very small or the loop has cycled a certain number of times.
- Lines 1008 – 1022 update the bead configuration from one time-point to the next, along with calculation of the distance between the bead configurations at two consecutive time-points.

B.5 properties.f90

This file contains the code for two subroutines used to calculate properties of the bead-spring chain over the course of a simulation.

- Lines 14 – 135 are for the `chain_props` subroutine. This calculates chain properties listed in the comments of lines 37 – 50 based on the bead configuration (`Rbead`) and the forces acting on the beads (`F`).
- Lines 139 – 187 are for the `time_correl` subroutine. This estimates the time correlations between the bead configurations at two different time-points.

Appendix C

Alterations to the code to incorporate the FENE-Fraenkel spring

To incorporate the FENE-Fraenkel spring force law into the existing code, several alterations have been made. Most of these changes fall into three categories:

- The addition of a new parameter to incorporate the non-zero natural length Q_0 of a FENE-Fraenkel spring (compared to the "zero natural length" of a FENE spring
- The addition of the FENE-Fraenkel spring force law and its associated properties, such as the spring potential ϕ.
- Existing variables made to store double-precision numbers in Fortran (whereas they were previously "single-precision"). This is due to the added sensitivity of the spring force to small changes in spring length, as any small deviations are now compared against the sum of the natural length and maximum extensibility, which is a much larger value than the maximum extensibility alone, as in the FENE spring case. This sensitivity can be seen by how close the roots of the cubic (Equation (2.7.22)) are to each other. Other variables have been changed to double-precision, to allow functions and subroutines to communicate with each other without a loss in precision.

Furthermore, the `sqrtb` variable that was used to denote $\sqrt{b}$ for the FENE case is used for the δ extensibility parameter in the FENE-Fraenkel case, as the two are analogous to each other.

C.1 sensemble.f90

- Lines 52 – 58 — The list of variables has been split into two groups. Most of the variables have now been set to double-precision. The remaining, single-precision variables are `t1rouse`, `t1zimm`, `emax` and `tol(MaxNDT)`.
- Lines 60 – 62 — The list of variables here are now set to double precision.
- Lines 67 – 68 — The list of variables has been split. Variable `sqrtb` (which holds the value of δ in the FENE-Fraenkel case, or $\sqrt{b}$ in the FENE case) is now double precision and a new, real-valued, double-precision variable `naturallength` is added to call the natural length, Q_0.
- Line 87 – The `naturallength` is added to this line, so that the natural spring length can be read from the `inputc.dat`.
- Line 208 — The word "`natlength`" is printed in the `result.dat` file. This will be where the natural spring length value, Q_0, as given by the variable `naturallength`, will be printed to confirm the right value has been read by the program.
- Lines 218 – 223 — To account for the extra variable, `naturallength`, being printed in the `result.dat` file, some of the formatting instructions for the file are modified to read as follows.

```
Write (resunit,202) (i, i=1,17)
     !16 CHANGED TO 17 TO MATCH NUMBER OF OUTPUTS
200  Format ('#',7(G12.0,2X),5(G12.0,2X, 'Error       ',2X))
     !6 CHANGED TO 7 TO MATCH NUMBER OF OUTPUTS
202  Format ('#',17(I2,12X)) !approx LJustification
     !16 CHANGED TO 17 TO MATCH NUMBER OF OUTPUTS
end If
```

- Lines 304 – 305 — The call of the `Initial_position` subroutine now includes the extra variable, `naturallength`.
- Lines 318, 326 and 334 — The fourth input for the `Time_Integrate_Chain` subroutine has been changed from "0." to "0.D0" to make it double-precision. This is to satisfy further changes made to the subroutine to give double-precision outputs.

- Lines 319, 327 and 335 — The call of the `Time_Integrate_Chain` subroutine now includes the extra variable, `naturallength`.
- Line 505 — Addition of the `naturallength` variable to list of outputs in the `result.dat` file
- Line 509 — An alternative to formatting line 508 for values printed in the `result.dat` file is provided here to give values of <Rˆ2>, the average squared distance between the first and last beads of a chain, and <Rgˆ2>, the chain radius-of-gyration, to a higher degree of precision.

```
!210     Format (I3, 11X, 6(G12.5,2X), 4(F20.5,2X), 6(G12.5,2X))
```

C.2 modules.f90

Changes in this module mainly correspond to the additional parameters and variables required to accommodate the spring natural length Q_0 into calculations performed by various subroutines and functions.

- Lines 25 – 26 — We define a new integer parameter `FENEFraenkel = 5`. This will be used to call up the FENE-Fraenkel spring force law in the code. The other parameters correspond to the Hookean, FENE, inverse Langevin chain and worm-like chain spring force laws.
- Lines 65 & 70 — The call of the `Initial_position` subroutine now includes an additional variable, `Q0s`, which will hold the natural spring length, Q_0. This variable is declared as a real-valued, double-precision input variable.
- Line 69 — Within the `Initial_position` subroutine, the variable `L0s`, which holds the value of δ in the FENE-Fraenkel case (or $\sqrt{b}$ in the FENE case), is now double-precision.
- Line 72 — Variable `R`, which stores the bead coordinates in the `Initial_position` subroutine, is now double-precision.
- Line 82 — Variable `GX`, which stores the Gaussian random vector created by the `GaussRand` subroutine, is now double-precision.

- Lines 91 & 106 — The call of the `Time_Integrate_Chain` subroutine now includes an additional variable, `Q0s`, which will hold the value of the natural spring length, Q_0. This variable is declared as a real-valued, double-precision, input variable.
- Line 96 — Variable `R_bead`, which stores the coordinates of the beads in a chain for the `Time_Integrate_Chain` subroutine, is now double-precision.
- Line 99 — A comment has been added to indicate that the `spring_force` variable can now use the FENEFraenkel parameter defined in lines 25 – 26 to implement the FENE-Fraenkel spring force law.
- Line 101 — Time related variables `tcur`, `tmax` and `delts` are now double-precision.
- Line 105 — Within the `Time_Integrate_Chain` subroutine, the variable `L0s`, which holds the value of δ in the FENE-Fraenkel case (or $\sqrt{b}$ in the FENE case), is now double-precision.
- Lines 111 – 112 — Variables `times` and `samples` (connected with the time-points at which the chain properties are to be measured during a simulation) are now double-precision.
- Line 137 — Variable `R(n)`, the random vector created by the `ran_1` subroutine, is now double-precision.
- Line 145 — Variables `A`, `maxev` and `minev` in the `maxminev_fi` subroutine are now double-precision.
- Line 153 — Variables `d_a`, `d_b` and `a` in the `chbyshv` subroutine are now double-precision.
- Line 171 – 172 — Variables `Rbead`, `F` and `props` in the `chain_props` subroutine are now double-precision.
- Lines 182 – 184 — Variables `R`, `F`, `correl`, `t0` and `t` in the `time_correl` subroutine are now double-precision.
- Lines 192 – 193 — Variables `f` and `dx` in the `numint` subroutine are now double precision. We cannot use `DBprec` to call double-precision here as the `bspglocons` module is not called here.

- Lines 208 & 211 — The call of the `normeqpr` subroutine now includes an additional variable, `natscaled`, which will hold the value $Q_0/s = q_0$. This variable is declared as a real-valued, double-precision, input variable.
- Lines 228 – 229 — Variables `t` and `K` in the `get_kappa` subroutine are now double precision.

C.3 utils.f90

- Lines 14 & 25 — The call of the `Initial_position` subroutine now includes an additional variable, `Q0s`, which will hold the value of the natural spring length, Q_0. This variable is declared as a real-valued, double-precision, input variable.
- Line 17 — `Use bspintfacs` now reads

  ```
  Use bspintfacs, IP_from_bspintfacs => Initial_position
  ```

  This avoids the issue where a compiler tries to compile a subroutine that has an interface with the same name coded for in the `bspintfacs` module of the `modules.f90` file.
- Line 24 — Within the `Initial_position` subroutine, the variable `L0s`, which holds the value of δ in the FENE-Fraenkel case (or $\sqrt{b}$ in the FENE case), is now double-precision.
- Line 26 — Variable `R`, which stores the bead coordinates in the `Initial_position` subroutine, is now double-precision.
- Lines 32 & 48 — A new, real-valued, single-precision variable, `natscaled`, is declared and created to store the scaled natural length $q_0 = Q_0/s =$ `L0s/Q0s`.
- Lines 60 – 86 — These lines now appear as follows.

```
if (Stype .ne. HOOK) then
 done = 0;
 do while (done .ne. 1)
    if (Stype .eq. FENEFraenkel) then !FENE-Fraenkel Case
        rv = (2.0 * ran()) - 1.0 + natscaled
```

```
        !Random number for the proportion of extension.
        !Add natscaled so that
        !rv = Q/s = Q/sqrt(b) = random + (Q0/s)
        else !Original case as in old code
            rv = ran()
        end if
        eqpr = normeqpr(Stype,rv,b,natscaled);
                !FENE-Fraenkel Replacement

        ! use another uniform random number to decide acceptance
        ! ie, accept only eqpr fraction at this r
        ! If variance reduction is not being used then simply use
                 ran() rather than ran(varseed)
        rvchk = ran()
        !write (*,*) 'rv = ', rv
        !write(*,*) 'eqpr = ', eqpr
        !write(*,*) 'rvchk = ', rvchk
        if (rvchk .le. eqpr/upbound) done = 1
    end do
    modr = Sqrt(Sum(b2b*b2b))
    b2b = b2b/modr * (rv * L0s)
    !b2b/modr gives a random direction for a unit vector
                !FENE-Fraenkel Replacement
    !(rv * L0s) = Q !FENE-Fraenkel Replacement
 end if
```

This part of the code is a Metropolis-Hastings rejection sampling algorithm that uses two uniform random variables; `rv` is the candidate length between beads after division by the maximum extensibility, and `rvchk` is used to accept or reject the value. For the other spring types used by this code, the spring length can vary between 0 and the maximum extensibility (*e.g.* for FENE springs, this is $\sqrt{b}$). Thus the variable `rv` can be between 0 and 1.

For the FENE-Fraenkel spring, we require the spring length Q to be in the range

$(Q_0 - s, Q_0 + s)$. Thus, `rv`, equal to the length divided by the maximum extensibility, needs to be between `natscaled`−1 and `natscaled`+1. The code is modified to reflect this setup. The function `normeqpr` now also has an additional input — `natscaled`.

- Line 104 — `Use bspintfacs` now reads

  ```
  Use bspintfacs, GR_from_bspintfacs => GaussRand
  ```

  This avoids the issue where a compiler tries to compile a subroutine that has an interface with the same name coded for in the `bspintfacs` module of the `modules.f90` file.

- Lines 109 – 112 — Variables `GX`, `rnd(2)`, `x`, `y`, `rsq` and `fac` in the `GaussRand` subroutine are now all double-precision.

- Line 141 — Variable `X(n)` in the `ran_1` subroutine is now double-precision. (In the `modules.f90` file, this variable is called `R(n)`.)

- Line 184 — Variables `d_a`, `d_b` and `a` in the `chbyshv` subroutine are now double-precision.

- Line 228 — Variables `A`, `maxev` and `minev` in the `maxminev_fi` subroutine are now double-precision.

- Line 234 — Variables `F(n)`, `G(n)`, `alpha` and `beta` in the `maxminev_fi` subroutine are now all double-precision. Furthermore, `ddot` replaces `sdot` as we require use of the `ddot`, double-precision variant of the `sdot`, vector dot product function in the BLAS fortran library.

- Lines 236 – 241 — Values of `F`, `G`, `alpha` and `beta` are now of the double-precision type. This is indicated in Fortran by the use of a `D` instead of `E` when quoting values in standard form.

- Lines 245 – 257 — BLAS subroutine `dsymv` and function `ddot` are used in place of their single-precision variants, `ssymv` and `sdot`.

- Lines 247 – 258 — Various values converted to double precision type.

- Lines 318 – 319 — Variables `f` and `dx` in the `numint` subroutine are now double-

precision. We cannot use `DBprec` to call double-precision here as the `bspglocons` module is not called here.

- Lines 383 & 387 — The `normeqpr` function now calls an additional variable, `natscal`. This is defined to be a real-valued, single-precision, input variable, and is used to store the scaled natural length $q_0 = Q_0/s$ = `L0s/Q0s` for the FENE-Fraenkel spring.

- Line 384 — The `bspglocons` module is called here, to allow the subroutine to change behaviour when the FENE-Fraenkel spring force law is in use, as the spring length range is different from those corresponding to other spring types.

- Lines 441 – 465 — These lines now appear as follows.

```
     if (Stype .eq. FENEFraenkel) then !FENE-Fraenkel Case
      do n=1,Ln
          rg = (X(n) * xfact) + natscal
          norm_b = norm_b +
                    W(n) * expphi(Stype,rg,b,natscal) * rg * rg
          fb = fb + W(n) * expphi(Stype,rg,b,natscal) * rg**4
      end do
   else !Original case as in old code
      do n=1,Ln
          rg = (X(n) + 1)/2 * xfact
          norm_b = norm_b +
                    W(n) * expphi(Stype,rg,b,natscal) * rg * rg
          fb = fb +  W(n) * expphi(Stype,rg,b,natscal) * rg**4
      end do
   end if !FENE-Fraenkel Replacement

   !write (*,*) '# b = ', b
   !write (*,*) '# computed normb = ', norm_b
   !write (*,*) '# f(b) = ', b*fb/norm_b

   bsav = b
```

```
   norm_computed = 1
end if

normeqpr = r*r*expphi(Stype,r,b,natscal)/norm_b
                !FENE-Fraenkel Replacement

end function normeqpr
```

This part of the code uses the potential function ϕ to calculate a probability distribution function based on $\exp(-\phi)$. This is discussed in more detail in Subsection 2.4.1. Part of the process involves integrating over the range of possible spring lengths — this is done here using a Gaussian quadrature with 21 quadrature points (lines 409 – 429).

As for lines 60 – 86, changes are made here to account for the fact that the range of possible lengths of a FENE-Fraenkel spring is different to that for springs obeying the other force laws used by this code. Furthermore, the function `expphi` has an additional input, `natscal`, which holds the value of the scaled natural spring length, $q_0 = Q_0/s$.

- Lines 468 & 473 — The function `expphi` now has an additional variable, `natlensc`, which stores the scaled natural spring length q_0. This variable is defined to be a real-valued, single-precision, input variable.

- Lines 484 – 485 — Outputs the potential function for the FENE-Fraenkel spring force law, as per Equation (2.4.6). This function stores values of $\exp(\phi)$ rather than $\exp(-\phi)$, but the calculations performed by the rest of the code correct for this change.

- Lines 513 – 514 — Variables `t` and `K` in the `get_kappa` subroutine are now double-precision.

- Lines 526 – 528 — Values for the κ matrix corresponding to uniaxial extensional flow have now been corrected. They are $\kappa(1,1) = 1$ (previously 0), $\kappa(2,2) = -0.5$ (previous 2) and $\kappa(3,3) = -0.5$ (previously 0).

C.4 gsipc.f90

- Lines 30 – 31 — Variables `R` & `b2b` in the `b2bvector_sym_up` subroutine are now double-precision.
- Lines 58 – 59 — Variables `b2bvec` and `deltaR` in the `modr_sym_up` are now double-precision.
- Lines 89 – 91 — Variables `alpha`, `vec` and `tensor` in the `tensor_prod_of_vec` subroutine are now double-precision.
- Line 113 — Variable `sqrtb`, representing the maximum extensibility parameter ($\sqrt{b}$ in the FENE case, and δ in the FENE-Fraenkel case), is now double-precision. A new, real-valued, double-precision variable, `natlength`, to store the scaled natural length q_0, is also declared here.
- Lines 122 – 153 — These lines now appear as follows.

```
Subroutine force_sans_hookean(sptype,r,ff)
                !FENE-Fraenkel Replacement
  Integer, Intent (in) :: sptype

  Real (DBprec), Intent (out) :: ff

  Real (DBprec) r
  Real (DBprec) scalednat !FENE-Fraenkel Replacement

  !r = |Q| / sqrtb  !COMMENT BY PRAVEEN
  scalednat = natlength/sqrtb ! = |Q0| / sqrtb
                !FENE-Fraenkel Replacement
  ! SCALED (by root b) natural spring length
                for the FENE-Fraenkel case.

  ! compute the force factor, ff
  Select Case (sptype)
  Case (HOOK) ! Hookean
```

```
      ff = 1.0

   Case (FENE) ! Warner Spring
      ff = 1./(1 - r*r)

   Case (ILC)  ! Inverse Langevin, Pade approx
      ff = (3-r*r)/(1 - r*r)/3.

   Case (WLC) ! Worm-like Chain, Marko-Siggia interpolation
      ff = 1./(6*r) * ( 4*r+ 1./(1-r)/(1-r) - 1)

   Case (FENEFraenkel) ! FENE-Fraenkel Spring
      ff = (1.-(scalednat/r))/(1.-((r-scalednat)*(r-scalednat)))
   !FENE-Fraenkel Replacement

   End Select
End Subroutine force_sans_hookean
```

This section of code calculates the spring connector force depending on the spring length.

- In the original code, the spring length was converted into double precision as partof the subroutine (`r = sr`). This line is now unneccessary, as is the sr variable, as r will now always be a double-precision input.
- For the FENE-Fraenkel spring, we also need to introduce a real, double-precision variable, `scalednat` which contains the scaled natural spring length q_0. This is not passed into the subroutine via the input variables, but by the variable natlength saved externally.
- Finally, the FENE-Fraenkel spring force law has its own associated expression (see Subsection 2.2.3 for derivation), which has been added as a fifth case to the `Select Case` part of the subroutine.

- Lines 158 – 160 — Variables `b2bvec`, `dr` and `Fs` in the `Spring_force` subroutine are now double-precision.

- Lines 194 – 327 — The `solve_implicit_r` subroutine code now appears as follows.

```
  !FENE-Fraenkel Replacement BELOW
Subroutine solve_implicit_r(sptype,dtby4,gama,natscl,r,ff,r_old)
  Integer, Intent (in) :: sptype
  Real (DBprec), Intent (in) :: dtby4, gama, natscl
  Real (DBprec), Intent (out) :: r
  Real (DBprec), Intent (out) :: ff

  Complex (DBprec) :: Solutions(3)
  Complex (DBprec) :: zA, zB, zC
  Real (DBprec) :: rA, rB, rC
  Real (DBprec), Intent (in) :: r_old
  Real (DBprec) :: dif

  !! Purpose
  ! solves the equation r ( 1 + dt/4 ff ) == gama
  ! where ff depends on spring force law F = H Q ff,
  ! for r and returns, r and ff(r)

  Real (DBprec) :: coeff(4)
  Real (DBprec) :: denom

  ! set up the polynomial equation (cubic), and the guess for
  ! gama >> 1, obtained by the asymptotic behaviour

  coeff(4) = 1.D0

  Select Case (sptype)
  Case (HOOK) ! Hookean
     r = gama/(1.D0+dtby4)
     ff = 1.D0
```

```
   Return

Case (FENE) ! FENE
   coeff(1) = gama
   coeff(2) = -(1.D0 + dtby4)
   coeff(3) = -gama
   r = 1.D0 - dtby4/2.D0/gama

Case (ILC) ! ILC
   denom = 3.D0 + dtby4
   coeff(1) = 3.D0 * gama / denom
   coeff(2) = -(3.D0 + 3.D0* dtby4) / denom
   coeff(3) = -3.D0 * gama / denom
   r = 1.D0 - dtby4/3.D0/gama

Case (WLC) ! WLC
   denom = 2.D0*(1.5D0 + dtby4)
   coeff(1) = -3.D0 *gama/denom
   coeff(2) =  3.D0 * (1.D0 + dtby4 + 2.D0*gama) / denom
   coeff(3) = -1.5D0 * (4.D0 + 3.D0*dtby4 + 2.D0*gama) / denom
   r = 1.D0 - Sqrt(dtby4/6.D0/gama)

Case (FENEFraenkel) ! FENE-Fraenkel
   coeff(1) = (dtby4*natscl) + gama - (gama*natscl*natscl)
   coeff(2) = (-1.D0)
              + (natscl*natscl) - dtby4 + (2.D0 * gama * natscl)
   coeff(3) = (-2.D0 * natscl) - gama
   Solutions = CubicSolver(coeff)
   zA = Solutions(1)
   zB = Solutions(2)
   zC = Solutions(3)
!(FENE Fraenkel uses CubicSolver AND double precision
```

```
      numbers used)

End Select

!write (*,*) "Gamma value calculated is ", gama
!write (*,*) "Polynomial is x^3 + ", coeff(3), "x^2 + ",
                  coeff(2), "x + ", coeff(1)
!write (*,*) "with dtby4 equal to", dtby4
!write (*,*) "First guess for r is ", r
!write (*,*) "The solver's numbers are ", zA, zB, zC

! the Hookean guess is same for all the force laws
If (gama < 1.0) Then
   r = gama/(1.+dtby4)
End If

! all the common forces laws yeild a cubic in the implicit form
! polish the guessed root by a Newt-Raph for polynomials

! for WLC, the newt raph, does not converge to the correct root,
! for gama > 100 , therefore simply ignore polishing
If (sptype .Eq. WLC .And. gama > 100) Then
   r = r
Else if (sptype .eq. FENEFraenkel) Then
   !Call polish_poly_root(coeff,rA,1e-6)
   !Call polish_poly_root(coeff,rB,1e-6)
   !Call polish_poly_root(coeff,rC,1e-6)
   !write (*,*) "Roots after polishing are ", rA, rB, rC

   r = 0.D0
   if (abs(imagpart(zA)) .lt. 1d-6) then
      rA = realpart(zA)
```

```
else
   rA = 3.D0*natscl + 1.D0
end if
if (abs(imagpart(zB)) .lt. 1d-6) then
   rB = realpart(zB)
else
   rB = 3.D0*natscl + 1.D0
end if
if (abs(imagpart(zC)) .lt. 1d-6) then
   rC = realpart(zC)
else
   rC = 3.D0*natscl + 1.D0
end if
!write (*,*) "The solver's real solutions are ", rA, rB, rC
if ((rA .le. (natscl - 1.D0)) .or.
           (rA .ge. (natscl + 1.D0))) then
   rA = 3.D0*natscl + 1.D0
end if
if ((rB .le. (natscl - 1.D0)) .or.
           (rB .ge. (natscl + 1.D0))) then
   rB = 3.D0*natscl + 1.D0
end if
if ((rC .le. (natscl - 1.D0)) .or.
           (rC .ge. (natscl + 1.D0))) then
   rC = 3.D0*natscl + 1.D0
end if

dif = min(abs(rA-r_old),abs(rB-r_old),abs(rC-r_old))
If (dif .eq. abs(rA-r_old)) then
   r = rA
Else If (dif .eq. abs(rB-r_old)) then
   r = rB
```

```
      Else If (dif .eq. abs(rC-r_old)) then
         r = rC
      End If
      if ((r .le. (natscl - 1.D0)) .or.
            (r .ge. (natscl + 1.D0))) then
         r = 1.D0*natscl + 1.D0
      end if
   Else
      Call polish_poly_root(coeff,r,1.D-6)
   End If
   !write (*,*) "Final guess for r is ", r
   Call force_sans_hookean(sptype,r,ff)
   !write (*,*) "Force_sans_hookean output is ", ff
End Subroutine solve_implicit_r
!FENE-Fraenkel Replacement ABOVE
```

The main changes are:

- There is now an extra, real-valued, double-precision, input variable called `r_old`. This stores the spring length from the previous time-step and is used to ensure that the new spring length calculated here is not too different from the previous value if there is more than one possible valid candidate for the new spring length.
- Existing variables `dtby4`, `gama`, `coeff` and `denom` are now of double-precision type.
- There is a new, real-valued, double precision variable, `natscl`, which represents the scaled natural spring length q_0.
- All real numbers that are used to calculate the spring length have now been converted to double-precision type. This is to improve the accuracy of the cubic solver (used for the FENE-Fraenkel; see below) or Newton-Raphson method (used for all other springs).
- The addition of the FENE-Fraenkel case.

- Optional code (commented out here) that can be used to output values of variables at intermediate steps to check that the correct numbers are being calculated.
- For the FENE-Fraenkel case, roots of the cubic polynomial (see Subsection 2.7.5) are calculated directly, instead of using the Newton Raphson method to hone in on a real root of the cubic polynomial (see change for lines 329 – 402 below).
    1. The roots can be "polished" by the Newton-Raphson process, but this has been commented-out of the code here.
    2. Complex roots are ignored, by resetting them to a value outside of the valid range of spring lengths.
    3. If the real roots being considered are out of the valid range, they are set to values outside of the valid range of spring lengths.
    4. If multiple roots lie in the valid range, then the one closest to the spring's existing length is chosen.
    5. If no roots lie in the valid range, then the chosen root is set to be equal to $q_0 + 1$, which will result in an infinite spring force being calculated, and causes an error.

- Lines 329 – 402 — A new function, `CubicSolver`, has been added:

```
  !FENE-Fraenkel Replacement INSERTION BELOW
Function CubicSolver(coeff)
!This cubic solver is based on that found in Section 5.6 of
!Numerical Recipes in Fortran: The Art of Scientific Computing,
!2nd edition (1992), Press et al.
!
!Alternative available at:
!http://blogs.warwick.ac.uk/bibojiang/entry/
            cubic_funtion_computational/
!http://www-old.me.gatech.edu/energy/andy_phd/appA.htm
```

```
Real (DBprec), Intent (in) :: coeff(4)
Real (DBprec) :: c(4)
Complex (DBprec) :: CubicSolver(3)
Complex (DBprec) :: x1, x2, x3

Real (DBprec), Parameter :: pi = 3.14159265358979323846D0

Real (DBprec) :: Q, R, diff

Real (DBprec) :: theta, sqrtQ

Real (DBprec) :: A, B
Real (DBprec) :: sqrt3by2
Real (DBprec) :: impart
Real (DBprec) :: repart1, repart2

c(1) = coeff(1)/coeff(4)
c(2) = coeff(2)/coeff(4)
c(3) = coeff(3)/coeff(4)
c(4) = 1.D0

Q = ((c(3)*c(3)) - (3.D0*c(2))) / 9.D0
!write(*,*) "Q is", Q
R = ((2.D0*c(3)*c(3)*c(3)) - (9.D0*c(2)*c(3)) +
                  (27.D0*c(1))) / 54.D0
!write(*,*) "R is", R
diff = (R*R) - (Q*Q*Q)

if (diff .le. 0.) then
   theta = acos(R/(sqrt(Q*Q*Q)))
   !write(*,*) "R/sqrt(Q3) is", R/(sqrt(Q*Q*Q))
   !write(*,*) "theta is", theta
```

```
   sqrtQ = sqrt(Q)
   x1 = complex((-2.D0 * sqrtQ *
         cos(theta/3.D0) - (c(3)/3.D0)),0.D0)
   x2 = complex((-2.D0 * sqrtQ *
         cos((theta + (2.D0*pi))/3.D0) - (c(3)/3.D0)),0.D0)
   x3 = complex((-2.D0 * sqrtQ *
         cos((theta + (4.D0*pi))/3.D0) - (c(3)/3.D0)),0.D0)
else
   !write(*,*) "Imaginary roots present!"
   A = sign(1.D0,R)*(((abs(R)) + sqrt(diff))**(1.D0/3.D0))
   B = 0.D0
   if (A .ne. 0.D0) B = Q/A
   sqrt3by2 = sqrt(3.D0)/2.D0
   repart1 = (A + B) - (c(3)/3.D0)
   repart2 = ((-0.5D0)*(A + B)) - (c(3)/3.D0)
   impart = sqrt3by2 * (A - B)
   x1 = complex(repart1, 0.D0)
   x2 = complex(repart2, impart)
   x3 = complex(repart2, (-1.D0*impart))
end if

CubicSolver(1) = x1
!write(*,*) "x1 is ", x1
CubicSolver(2) = x2
!write(*,*) "x2 is ", x2
CubicSolver(3) = x3
!write(*,*) "x3 is ", x3

!VERIFICATION
!write(*,*) "Cube root of 27 is", 27.**(1./3.)
!write(*,*) "Plugging in the solver values gives ",
        (coeff(4)*x1*x1*x1) + (coeff(3)*x1*x1) &
```

```
!+ (coeff(2)*x1) + (coeff(1)), (coeff(4)*x2*x2*x2) +
        (coeff(3)*x2*x2) + (coeff(2)*x2) + (coeff(1)), &
!(coeff(4)*x3*x3*x3) + (coeff(3)*x3*x3) +
        (coeff(2)*x3) + (coeff(1))

End Function CubicSolver
!FENE-Fraenkel Replacement INSERTION ABOVE
```

For the four other spring types simulated, a solution to the cubic polynomial (see Subsection 2.7.5) is found by making an initial guess and use of the Newton-Raphson method to polish the root. Work with the FENE-Fraenkel spring force law and this method has shown that the initial guess proposed can greatly affect which root you converge towards. Therefore, we use a cubic solver function based on an algorithm in Press et al. [115, §5.6] to directly calculate the roots of the cubic polynomial, so that a valid root can be determined elsewhere.

The procedure is as follows. For a cubic equation

$$x^3 + ax^2 + bx + c = 0 \tag{C.4.1}$$

with real (or complex) coeficients a, b, c, first compute:

$$Q = \frac{a^2 - 3b}{9} \quad \text{and} \quad R = \frac{2a^3 - 9ab + 27c}{54} \tag{C.4.2}$$

If Q and R are real and $R^2 < Q^3$, then the cubic equation Equation (C.4.1) has three real roots:

$$x_i = -2\sqrt{Q}\cos\left(\frac{\theta + 2\pi(i-1)}{3}\right) - \frac{a}{3} \tag{C.4.3}$$

where

$$\theta = \arccos\left(\frac{R}{\sqrt{Q^3}}\right) \tag{C.4.4}$$

and $i = 1, 2, 3$.

If at least one of Q and R is not real, or if $R^2 \geq Q^3$, we then need to calculate:

$$A = -\left[R \pm \sqrt{R^2 - Q^3}\right]^{1/3}, \tag{C.4.5}$$

where the $\pm$ is chosen so that $\Re(R^* \times \pm\sqrt{R^2 - Q^3}) \geq 0$, with R^* denoting the complex conjugate of R. Next, we calculate:

$$B = \begin{cases} Q/A & (A \neq 0) \\ 0 & (A = 0). \end{cases} \tag{C.4.6}$$

Then the three roots of the cubic are:

$$\begin{aligned} x_1 &= (A + B) - \frac{a}{3} \\ x_2 &= -\frac{1}{2}(A + B) - \frac{a}{3} + i\frac{\sqrt{3}}{2}(A - B) \\ x_3 &= -\frac{1}{2}(A + B) - \frac{a}{3} - i\frac{\sqrt{3}}{2}(A - B) \end{aligned} \tag{C.4.7}$$

The code provides for all eventualities. It also includes lines (commented out here) that can be used to diagnose if the code is calculating values incorrectly.

- Lines 409 – 411 — Variables `b2bvec`, `dr` and `Fev` in the `Excluded_volume_force` subroutine are now double-precision.

- Lines 445 – 449 & 473 — The call of the `Time_Integrate_Chain` subroutine now includes an additional variable, `Q0s`, which will hold the value of the natural spring length, Q_0. This variable is declared as a real-valued, double-precision, input variable.

- Line 453 — `Use bspintfacs` now reads

  ```
  Use bspintfacs, TIC_from_bspintfacs => Time_Integrate_Chain
  ```

  This avoids the issue where a compiler tries to compile a subroutine that has an interface with the same name coded for in the `bspintfacs` module of `modules.f90`.

- Lines 462, 468, 470, 472, 473, 478 and 479 — Variables `R_bead`, `tcur`, `tmax`, `delts`, `Hstar`, `L0s`, `times` and `samples` in the `Time_Integrate_Chain` subroutine are now all double-precision.

- Lines 481 – 482 — Two new, real-valued, double-precision variables, `R_trvec (Ndim)` and `rtrial`, are declared. These are used to calculate and store a spring's connector vector and length from the previous time-step, with roots of the cubic polynomial from the `solve_implicit_r` subroutine being compared against these values.
- Line 497 — Two new, real-valued, double-precision variables, `dnrm2` and `ddot`, are declared here to store values obtained from BLAS functions `dnrm2` (calculates the length of a vector) and `ddot` (calculates the dot product of two vectors) are used in place of their single-precision variants, `snrm2` and `sdot`.
- Lines 499 – 530 — All variables in these lines are now double-precision.
- Lines 541 – 543 — The variable `r` has been moved to a separate line and converted to double-precision.
- Lines 546 – 548 — The double-precision BLAS subroutine `dsymv` replaces the single-precision variant, `ssymv`, and variables `alpha` and `beta` (used in the routine) are now double-precision.
- Line 551 — The value of `natlength` is defined here — `natlength = Q0s`.
- Lines 643 – 644 — Two `write(*,*)` lines have been added to output the values of `F_spring` and `F_ev`, for code-checking purposes.
- Lines 747 – 760 — Throughout this section of code, the double-precision BLAS subroutine `dsymv` replaces the single-precision variant, `ssymv`, and variables `alpha` and `beta` (used in the routine) are now double-precision.
- Lines 777 – 786 — Throughout this section of code, the double-precision BLAS subroutine `dsymv` and function `ddot` replace their single-precision variants, `ssymv` and `sdot`. Furthermore, variables `alpha` and `beta` (used in the routine) are now double-precision.
- Lines 822 – 825 — The double-precision BLAS subroutine `dsymv` replaces the single-precision variant, `ssymv`, and variables `alpha` and `beta` (used in the routine) are now double-precision.
- Lines 859 – 861 — The double-precision BLAS subroutine `dsymv` replaces the

single-precision variant, `ssymv`, and variables `alpha` and `beta` (used in the routine) are now double-precision.

- Line 814 — A `write(*,*)` line has been added to output the value of `DelS`, for code-checking purposes.
- Line 841 — A `write(*,*)` line has been added to output the value of `R_bead` (the original bead configuration), for code-checking purposes.
- Line 906 — `dtsby4 = Delts/4.0` has been changed to `dtsby4 = Delts/4.0D0`, so that `dtsby4` is a double-precision value.
- Line 924 — A `write(*,*)` line has been added to output the value of `F_con_mu` (the spring connector force), for code-checking purposes.
- Lines 936 – 937 — Two `write(*,*)` lines have been added to output the values of `F_spring(:, mu)` and `F_spring(:,mu+1)`, for code-checking purposes.
- Line 942 — `gama_mag = snrm2(Ndim,gama_mu,1) !  BLAS single normal two` has been changed to

```
gama_mag = dnrm2(Ndim,gama_mu,1) !FENE-Fraenkel Replacement
                ! BLAS DOUBLE normal two
```

  Since `gama_mag` now needs to be double precision for compatibility, we need to use `dnrm2` and not `snrm2`.

- Lines 944 – 974 — This now reads as follows.

```
!FENE-Fraenkel Replacement BELOW
R_trvec = R_bead(:,mu+1) - R_bead(:,mu)
!write (*,*) "R_trvec is ", R_trvec
rtrial = (dnrm2(Ndim,R_trvec,1))/sqrtb
!write (*,*) "Original value of r (trial) is ", rtrial
!FENE-Fraenkel Replacement ABOVE

! r = Q_nu_mag/sqrt(b) varies from (0,1)
Call solve_implicit_r(spring_type,dtsby4,gama_mag/sqrtb, &
     natlength/sqrtb,r,ff,rtrial) !FENE-Fraenkel Replacement
```

```
!Write(*,*) "Output of solve_imp_r is ff = ", ff
            !FENE-Fraenkel Replacement
!write(*,*) 'Press Enter to continue'
            !FENE-Fraenkel Replacement
!read(*,*) !FENE-Fraenkel Replacement
! implicit solution of Eq.(21)
!   r ( 1 + deltat/4 ff) - |gama_mu|/sqrtb == 0
! and returns r and ff, the factor in the spring force other
! than hookean F = H Q ff

if (spring_type .eq. FENEFraenkel) then !FENE-Fraenkel case
            !FENE-Fraenkel Replacement
   if ((r - (natlength/sqrtb)) .gt. (1. - (1e-6))) then
      r = (natlength/sqrtb) + 1. - 1e-6
   end if
   if ((r - (natlength/sqrtb)) .lt. ((1e-6) - 1.)) then
      r = (natlength/sqrtb) - 1. + 1e-6
   end if
else !All other cases (no natural length)
            !FENE-Fraenkel Replacement
   if (Abs(r-1.) .Lt. 1e-6) r = 1 - 1e-6
end if
!If value for r gets too close to limits of the range
        of allowable lengths,
!between (natlength/sqrtb) - 1 and (natlength/sqrtb) + 1,
!then force it within range.
```

- In lines 944 – 949, the calculation of the previous spring length is carried out. There are also `write(*,*)` lines to output the results for code-checking purposes.

- In lines 952 – 953, `solve_implicit_r` is called with the input list

  `(spring_type,dtsby4,gama_mag/sqrtb,natlength/sqrtb,r,ff)`

  - In line 954, there is a `write(*,*)` line to output the force computed, for code-checking purposes.
  - In lines 955 – 956, there is a combination of lines which help to instgiate a pause in the program, so that errors can be spotted when they occur and not at the end of the simulation, or when the code has crashed.
  - Line 962 is modified to take into account the new range of lengths permitted by a FENE-Fraenkel spring.

- Line 995 — The two instances of `snrm2` in this line have been changed to `dnrm2`, to reflect the fact that the variable `R_bead` and associated chain configuration vectors are now stored with double precision.
- Line 1006 — A `write(*,*)` line has been added to output the value of `R_corr` (the new bead configuration), for code-checking purposes.

C.5 properties.f90

- Lines 19 – 20 — Variables `Rbead`, `F` and `props` in the `chain_props` subroutine are now double precision.
- Lines 144 – 146 — Variables `R`, `F`, `correl`, `t0` and `t` in the `time_correl` subroutine are now double precision

Appendix D

Fortran code for simulations

D.1 sensemble.f90

```
! Time-stamp: <sensemble.f90 15:32, 03 May 2018 by DP Amarasinghe>


!!! $Log: sensemble.f90,v $
!!! Revision 1.3  2004/03/17 02:05:24  pxs565
!!! Working version of a sample demonstration
!!!
!!! Revision 1.2  2004/03/16 21:48:12  pxs565
!!! minor variations, tried using an ntdone file,
!!!
!!! Revision 1.1  2004/03/16 07:01:14  pxs565
!!! Initial revision
!!!


Program chainsim_p
  Use bspglocons
  Use bspglovars
  Use bspintfacs
  Use Flowvars
  Use Flock_utils
```

```
    Implicit None


!-----------------------------------------------------------
!        Other Declarations
!-----------------------------------------------------------


    Character (len=12) clk(3)
    Integer (k4b) nseed

    ! File i/o
    Character(10), parameter :: FormatVersion = "GAVG-1.0"
    !Character (10) :: fver
    Character (len=20) infile, outfile, xdfile, gavgfile, ntfile, ...
       contfile,fver

    Integer, Parameter ::   inunit=10, outunit=11, gavunit=12, &
         resunit=15, cunit=16
    Logical :: Filexists, Flocked

    ! for gfortran
    Character (len=80) :: Format3, Format31, Format4, Format42, &
         Format43, Format5, Format51, Format6

    ! Trajectories related
    Integer, Parameter :: MaxNDT = 10
    Integer i, j, clok(8)
    Integer nthis, nblock, iblock, ntrajdone, ntraj, Nsamples, &
         idelts, ndelts,  ntrajvals(MaxNDT), nthisvals(MaxNDT)

    !Real tlongest,  t1rouse, t1zimm, teqbm, emax, tmax, deltseq, ...
       deltsne, & !FENE-Fraenkel Replacement
    !     deltseqvals(10), deltsvals(MaxNDT), tol(MaxNDT) ...
       !FENE-Fraenkel Replacement
    !FENE-Fraenkel Replacement BELOW
    Real t1rouse, t1zimm, emax, tol(MaxNDT)
    Real (DBprec) tlongest, teqbm, tmax, deltseq, deltsne, &
                  deltseqvals(10), deltsvals(MaxNDT)
```

```
!FENE-Fraenkel Replacement ABOVE

Real (DBprec) , Allocatable :: PosVecR(:,:), times(:), ...
   samples(:,:), & !FENE-Fraenkel Replacement
     avgs(:,:), errs(:,:), &
     global_avgs(:,:), global_errs(:,:)


Integer SType,mmult, nsact, cidx, ord, eqprops, neqprops

Real hstar, zstar, dstar!, sqrtb !FENE-Fraenkel Replacement
Real (DBprec) sqrtb, naturallength
Real binwidth, intCss, csserr, l1rouse
Character (len=10), parameter :: ErString = "Error"
Real  rems, reerr, rgms, rgerr, xms, xerr, dfvty, derr, ntot, Conf_t


Integer :: NBeads
Real :: Delts

!-----------------------------------------------------------------
!        Get input data
!-----------------------------------------------------------------

infile="inputc.dat"


Open (unit=inunit,file=infile,status="old")
Read (inunit,*)
Read (inunit,*) SType, NBeads, hstar, zstar, dstar, sqrtb, &
      naturallength, Gdots !FENE-Fraenkel Replacement
Read (inunit,*)
Read (inunit,*) emax, Nsamples
Read (inunit,*)
Read (inunit,*) ndelts
Read (inunit,*)

If (ndelts > MaxNDT) Then
   Write(*,*) 'Number of delt exceeded ', MaxNDT
```

```
      Stop
   End If

   Do i = 1, ndelts
      Read(inunit,*) deltseqvals(i), deltsvals(i), nthisvals(i), &
           ntrajvals(i), tol(i)
   End Do
   Close (unit=inunit)

   !-------------------------------------------------------------
   !        Initialization
   !-------------------------------------------------------------
   If (Nsamples.Lt.2) Nsamples = 2

   Allocate(PosVecR(Ndim,NBeads))
   Allocate ( &
        times(Nsamples),  &
        samples(NProps,Nsamples), &
        avgs(NProps,Nsamples), &
        errs(NProps,Nsamples))

   Allocate(&
        global_avgs(NProps,Nsamples), &
        global_errs(NProps,Nsamples))

   global_avgs = 0
   global_errs = 0



   times = 0

   ! longest relaxation times Rouse and Zimm
   t1rouse =   0.5/Sin(PI/2/Nbeads)**2
   t1zimm = lam1_th(hstar,NBeads) ! use thurstons formula for Zimm

   tlongest = t1rouse
```

```
135   teqbm = tlongest ! without EV

      If (zstar .Ne. 0) Then
         teqbm = 3.0*tlongest ! with excluded volume, it takes roughly ...
            3 times
         ! to attain equilibrium even with init dist
140   End If

      If (gdots == 0) Then
         tmax = emax
      Else
145      tmax = emax/gdots
      End If

      !-------------------------------------------------------------
      !        Initialization variable format expressions
150   !         <> language extension not available in gfortran
      !-------------------------------------------------------------

      Write(Format3,"(a,I3,a)") "('#',", 2*NProps+4, "(A11,1X))"
      ! short cut way to get (nearly) left justification
155   Write(Format31,"(a,I3,a)"),  "('#',", 2*NProps+4,"(G2.0,10X))"



      If (gdots == 0) Then
160      eqprops = 4
      Else
         neqprops = 4
      End if
   !!$  Write(Format4 ,"(a,I3,a)") "(", 2*NProps+4,"(G11.4,1X))"
165   Write(Format42,"(a,I3,a)") "(", 2*eqprops+1, "(G11.4,1X))"
      Write(Format43,"(a,I3,a)") "(", 2*neqprops+2, "(G11.4,1X))"

   !!$  Write(Format5, "(a,I3,a)") "('#',A11,1X,", nsact, "(G11.4,2X))"
   !!$  Write(Format51, "(a,I3,a)") "('#',", nsact+1, "(I3,10X))"
170 !!$
   !!$  Write(Format6, "(a,I3,a)") "(", nsact+1, "(G11.4,2X))"
```

```
!-------------------------------------------------------------
!        Initialization of variables
!-------------------------------------------------------------


! the standard error of mean obtained from t-distribution
! for sample mean and sample standard deviation
Conf_t = 2 ! 95% confidence for degrees of freedom > 20



Prop_names(1) = "R^2"          ! square of end-to-end vector
Prop_names(2) = "S11"          ! 1,1 component of shape tensor
Prop_names(3) = "S12"          ! 1,2 component of shape tensor
Prop_names(4) = "S13"          ! 1,3 component of shape tensor
Prop_names(5) = "S22"          ! 2,2 component of shape tensor
Prop_names(6) = "S23"          ! 2,3 component of shape tensor
Prop_names(7) = "S33"          ! 3,3 component of shape tensor
Prop_names(8) = "N1"           ! First normal stress difference
Prop_names(9) = "N2"           ! Second normal stress difference
Prop_names(10) = "T12"         ! 1,2 component of stress tensor
Prop_names(11) = "X1"          ! Stretch in 1 direction
Prop_names(12) = "X2"          ! stretch in 2 direction
Prop_names(13) = "X3"          ! stretch in 3 direction
Prop_names(14) = "Rg^2"        ! sq. Radius of gyration
Prop_names(15) = "PC error"    ! Predictor-Corrector Error
Prop_names(16) = "<SxySxy>"    ! Auto correlation fn for shear stress
Prop_names(17) = "Diffusvty"   ! Center of mass diffusivity



If (gdots == 0 ) Then
   Open (unit = resunit, file = "result.dat", status="unknown")
   Write (resunit,200) "NBeads      ", &
        "sqrtb       ", &
        "natlength   ", & !FENE-Fraenkel Replacement
        "h*          ", &
        "z*          ", &
        "d*          ", &
```

```
            "dtsne       ", &
            "<R^2>       ", &
            "<Rg^2>      ", &
            "Diffsvty    ", &
            "l_eta       ", &
            "<x>         "
      Write (resunit,202) (i, i=1,17)
        !16 CHANGED TO 17 TO MATCH NUMBER OF OUTPUTS !FENE-Fraenkel ...
           Replacement
200   Format ('#',7(G12.0,2X),5(G12.0,2X, 'Error       ',2X))
        !6 CHANGED TO 7 TO MATCH NUMBER OF OUTPUTS !FENE-Fraenkel ...
           Replacement
202   Format ('#',17(I2,12X)) !approx LJustification
        !16 CHANGED TO 17 TO MATCH NUMBER OF OUTPUTS !FENE-Fraenkel ...
           Replacement
    end If



    !---------------------------------------------------------------
    !     Generate the seeds based on the current time
    !---------------------------------------------------------------

    Call Date_and_time(clk(1), clk(2), clk(3), clok)
    !       ms              sec         min        hr
    nseed = clok(8)*100000+clok(7)*1000+clok(6)*10+clok(5)

    !-- an unique seed incase clok is the same
    nseed = (nseed + 201271)



    !---------------------------------------------------------------
    !    Begin the loop for time step sizes
    !---------------------------------------------------------------

    timesteps: Do  idelts = 1,ndelts
```

```
      deltseq = deltseqvals(idelts)
      deltsne = deltsvals(idelts)
250   Imploop_tol = tol(idelts)

      mmult = Int(tmax/(Nsamples-1)/deltsne) + 1

      ! actual no of samples to b taken
255   nsact = tmax/deltsne/mmult + 1

      Do i = 1,nsact
         times(i) = (i-1)*mmult*deltsne
      End Do
260
      ! Obtain the number of traj completed from the disk, if present

      Write (gavgfile, '("gavgs.",I2.2)') idelts

265   Inquire (file=gavgfile, exist=Filexists)

      If (Filexists) Then
         open (unit=gavunit,file=gavgfile,status='old')

270      Read (gavunit,*) fver  ! Format version
         if (fver /= FormatVersion) Then
            Write (*,*) 'Incompatible Format version in ', gavgfile
            close(gavunit)
            Go to 99999
275      end if

         Read (gavunit,*) ntrajdone
         close(gavunit)


280
      Else
         ntrajdone = 0
      End If
      !--when several procs start at the same time it
285   !  can lead to the total trajs done being > ntrajvals
      ntraj = Max(0,ntrajvals(idelts)-ntrajdone)
```

```
      !--number of trajectories for this run
      nthis = Min(ntraj, nthisvals(idelts))

      nblock = nthis

write(*,*) 'nblock = ', nblock
      avgs = 0
      errs = 0
      !---------------------------------------------------------------
      !    Begin the loop for the blocks
      !---------------------------------------------------------------
      trajectories: Do iblock = 1, nblock
         samples = 0.0
         write (*,*) idelts, 'Traj = ', iblock

         PosVecR = 0

         Call ...
             Initial_position(SType,Nbeads,sqrtb,naturallength,PosVecR,nseed)
!FENE-Fraenkel Replacement
         ! debug

 ! PosVecR(1,1) =  -0.562280883445
 ! PosVecR(2,1) = 0.408563747373
 ! PosVecR(3,1) = -0.496354712261
 ! PosVecR(1,2) =  -1.279498896389
 ! PosVecR(2,2) = 0.325713351306
 ! PosVecR(3,2) = 1.423666808622


         FlowType = EQ
         Call Time_Integrate_Chain(NBeads, PosVecR, SType,  &
              0.D0, teqbm, deltseq, & !FENE-Fraenkel Replacement
              0. , zstar, dstar, sqrtb, naturallength, & ...
                 !FENE-Fraenkel Replacement
              nseed, 0, times, samples )

         ! If doing equilibrium studies, use a large deltseq=1 first
         ! and later use deltsne for gathering data
```

```
      If (zstar .Ne. 0) Then
         Call Time_Integrate_Chain(NBeads, PosVecR, SType,  &
              0.D0, 4*tlongest, deltsne, & !FENE-Fraenkel Replacement
              0. , zstar, dstar, sqrtb, naturallength, & ...
                 !FENE-Fraenkel Replacement
              nseed, 1, times, samples )
      End If


      FlowType = SH
      Call Time_Integrate_Chain(NBeads, PosVecR, SType,  &
           0.D0, tmax, deltsne, & !FENE-Fraenkel Replacement
           hstar, zstar, dstar, sqrtb, naturallength, & ...
              !FENE-Fraenkel Replacement
           nseed, nsact, times, samples )


      avgs = avgs + samples
      errs = errs + samples*samples

   End Do trajectories


   !-------------------------------------------------------------
   !    Consolidate and save final results
   !-------------------------------------------------------------

   ! obtain global values from the disk
   Inquire (file=gavgfile, exist=Filexists)
   If (Filexists) Then
      !-- check if it is locked
      Flocktest: do
         call islocked(gavgfile,Flocked)
         if(.not. Flocked) Then
            call lockfile(gavunit,gavgfile)
            !-- to be unlocked after new data is written

            open (unit=gavunit,file=gavgfile,status='old')
            Read (gavunit,*) fver  ! Format version
```

```
            if (fver /= FormatVersion) Then
               Write (*,*) 'Incompatible Format version in ', ...
                  gavgfile
               call unlockfile(gavunit,gavgfile)
               Go to 99999
            end if

            Read (gavunit,*) ntrajdone
            Read (gavunit,*) global_avgs(1,:) , global_errs(1,:)
            Read (gavunit,*) global_avgs(8,:) , global_errs(8,:)
            Read (gavunit,*) global_avgs(10,:), global_errs(10,:)
            Read (gavunit,*) global_avgs(11,:), global_errs(11,:)
            Read (gavunit,*) global_avgs(14,:), global_errs(14,:)

            Close (gavunit)
            Exit Flocktest
         Else  ! when File is locked
            ! Sleep for a while and try again until it is unlocked,
            ! requires -Vaxlib in ifc
            call sleep(1)
         end if
      end do Flocktest
   Else ! if Globalavgs does .not. Filexists

      ! lock the file b4 opening a new one for writing
      call lockfile(gavunit,gavgfile) ! tobe unlocked after new ...
         data is written
      global_avgs = 0
      global_errs = 0
      ntrajdone = 0

   end If


   !-- add the current runs to that in the disk
   global_avgs = global_avgs + avgs
   global_errs = global_errs + errs
   ntrajdone = ntrajdone + nblock
```

```
      ! take the sum-total of time-ensemble averages for
      ! steady equilibrium measurments before averaging
      ! over ensembles

      ntot = nsact * ntrajdone
      rems  = Sum(global_avgs(1,1:nsact))/ntot
      reerr = Conf_t *  &
           ((Sum(global_errs(1,1:nsact)) - &
           ntot * rems*rems)/ntot/(ntot-1))**0.5

      rgms  = Sum(global_avgs(14,1:nsact))/ntot
      rgerr = Conf_t *  &
           ((Sum(global_errs(14,1:nsact)) - &
           ntot * rgms*rgms )/ntot/(ntot-1))**0.5

      xms  = Sum(global_avgs(11,1:nsact))/ntot
      xerr = Conf_t *  &
           ((Sum(global_errs(11,1:nsact)) - &
           ntot * xms*xms )/ntot/(ntot-1))**0.5


      ntot = (nsact-1)*ntrajdone

      dfvty  = Sum(global_avgs(17,2:nsact))/ntot
      derr = Conf_t *  &
           ((Sum(global_errs(17,2:nsact)) - &
           ntot * dfvty*dfvty)/ntot/(ntot-1))**0.5

      ! average and errors over the ensemble
      global_avgs = global_avgs/ntrajdone
      If (ntrajdone.Gt.2) Then
         global_errs = ...
             Conf_t*(Abs(global_errs-ntrajdone*global_avgs* &
               global_avgs)/ntrajdone/(ntrajdone-1))**0.5
      End If

      Write (outfile, '("output.",I2.2)') idelts
      Open (unit = outunit, file = outfile, status="unknown")
      Write (outunit,1) "SpringTyp ", "NBeads    ", "sqrtb     ", &
```

```
          "h*        ", "z*        ", "d*        ", &
          "gdot*     ", "dt*eq     ", "dt*ne     ", &
          "imp-tol   ", "TLongest  ", "TEqbm     ", &
          "eMax      ", "Tmax      ", "NTraj     " ,&
          "FlowType  "
      Write (outunit,2) SType, NBeads, sqrtb, hstar, zstar, dstar, &
           gdots, deltseq, deltsne, Imploop_tol, tlongest, teqbm, &
           emax, tmax, ntrajdone, FlowType


      If (gdots .Eq. 0) Then
         eqprops = 4
         Write (outunit,Format3) "Time       ",  &
              Prop_names(1), Erstring, &
              Prop_names(14), Erstring, &
              Prop_names(17), Erstring, &
              Prop_names(11), Erstring
         Write (outunit,Format31) (i, i=1,1+eqprops*2)
         Write (outunit,Format42) (times(i), &
              global_avgs(1,i), global_errs(1,i), &
              global_avgs(14,i), global_errs(14,i), &
              global_avgs(17,i), global_errs(17,i), &
              global_avgs(11,i), global_errs(11,i), &
              i = 1,nsact )

      Else
         neqprops = 4
         Write (outunit,Format3) &
              "Time      ",  &
              "Strain    ",  &
              Prop_names(1), Erstring, &
              Prop_names(14), Erstring, &
              "etaPEF     ", Erstring, &
              "etaShr     ", Erstring
         Write (outunit,Format31) (i, i=1,2+neqprops*2)
         Write (outunit,Format43) (times(i), times(i)*gdots,  &
              global_avgs(1,i), global_errs(1,i), &
              global_avgs(14,i), global_errs(14,i), &
```

```
475               -global_avgs(8,i)/gdots, global_errs(8,i)/gdots, &
                  -global_avgs(10,i)/gdots, global_errs(10,i)/gdots, &
                  i = 1,nsact )
         End If


480
         If (gdots .Eq. 0) Then
            ! The integral of the ensenble averaged Stress-stress ...
               correlation
            ! function gives the intrinsic viscosity at zero shear rate
            cidx = 16
485
            Do ord=1,3
               Call numint(global_avgs(cidx,:), deltsne*mmult, 1, ...
                  nsact, &
                    ord, intCss)
               Call numint(global_errs(cidx,:), deltsne*mmult, 1, ...
                  nsact, &
490                 ord, csserr)
               Write (outunit,101)  ord , intCss, csserr + 1./nsact**ord
   101         Format ('# zero sh rate intrinsic visc, ord = ',I2, ':', &
                    F10.6,' +/- ',F10.6)
            End Do
495
            ! lambda_eta from Rouse model
            l1rouse = 0
            Do i=1,Nbeads-1
               l1rouse = l1rouse +  0.5/Sin(i*PI/2/Nbeads)**2
500         End Do
            Write (outunit,*) '# from Rouse model = ', l1rouse
            Write (outunit,219) t1rouse,t1zimm
   219      Format('#longest Rouse = ',G10.3, ', Zimm = ',G10.3)

505         Write (resunit,210) NBeads, sqrtb, naturallength, & ...
               !FENE-Fraenkel Replacement
                 hstar, zstar, dstar, deltsne, &
                 rems, reerr, rgms, rgerr, dfvty, derr, intCss, csserr, ...
                    xms, xerr
   !210     Format (I3,  11X, 16(G12.5,2X))
```

```
210     Format (I3, 11X, 6(G12.5,2X), 4(F25.8,2X), 6(G12.5,2X)) !TEST
510

        end If ! gdots = 0

        Close (unit = outunit)
515

        !-------------------------------------------------------------
        ! Save the global averages data on to the disk
        ! But before that revert to the unnormalised values
520     ! Note that the gavunit file is still locked
        !-------------------------------------------------------------
        Open(unit=gavunit,file=gavgfile,status='replace')

        If (ntrajdone.Gt.2) Then
525        global_errs = (global_errs/Conf_t)**2 * ntrajdone * ...
               (ntrajdone-1)&
                + ntrajdone * global_avgs**2
        end If

        global_avgs = global_avgs * ntrajdone
530

        Write (gavunit,*) FormatVersion
        Write (gavunit,*) ntrajdone
        Write (gavunit,*) global_avgs(1,:) , global_errs(1,:)
        Write (gavunit,*) global_avgs(8,:) , global_errs(8,:)
535     Write (gavunit,*) global_avgs(10,:), global_errs(10,:)
        Write (gavunit,*) global_avgs(11,:), global_errs(11,:)
        Write (gavunit,*) global_avgs(14,:), global_errs(14,:)
        Close (gavunit)

540     !-- global avgs file is released for use with other processors
        call unlockfile(gavunit,gavgfile)


        DeAllocate( global_avgs, global_errs)
545

        !-------------------------------------------------------------
```

```
      !           Write info to disk for continuation
      !-------------------------------------------------------------
      ntraj = ntrajvals(idelts)-ntrajdone

      Write (contfile, '("continue.",I2.2)') idelts
      open(unit=cunit,file=contfile,status='replace')

      if (ntraj > 0) Then
         Write (cunit,*) ntraj, '  more trajectories to be completed'
         close(cunit)
      Else
         close(cunit,status='delete')
      end if


  End Do timesteps

  Close(resunit)


  ! Common termination statements
99999   Stop
  !-------------------------------------------------------------
  !                    Format statements
  !-------------------------------------------------------------

1  Format ('#',16(A10,2X))
  !2  Format ('#', 2(I10,2X), 12(G10.3,2X), (I10,2X)  )
2  Format ('#', 2(G2.0,10X), 12(G11.4,1X), (G10.3,2X), G2.0,11X  )

!3 Format ('#', <2*NProps+4>(A11,1X))
!3  Format (Format3)
  ! short cut way to get (nearly) left justification
!31 Format ('#', <2*NProps+4>(G2.0,10X))
!31 Format (Format31)

!4 Format (<2*NProps+4>(G11.4,1X))
!42 Format (<2*eqprops+1>(G11.4,1X))
!43 Format (<2*neqprops+2>(G11.4,1X))
```

```
!4  Format (Format4)
!42 Format (Format42)
!43 Format (Format43)

!5 Format ('#',A11,1X,<nsact>(G11.4,2X))
!51 Format ('#', <nsact+1>(I3,10X))
!5  Format (Format5)
!51 Format (Format51)

!6 Format (<nsact+1>(G11.4,2X))
!6  Format (Format6)

72 Format (5(F10.4,2x))
80 Format (3(F10.4,2x))

End Program chainsim_p
```

D.2 modules.f90

```
!!! Time-stamp: <modules.f90 15:32, 03 May 2018 by DP Amarasinghe>

!-----------------------------------------------------------------
!   Global modules and interface headers
!-----------------------------------------------------------------

!!! $Log: modules.f90,v $
!!! Revision 1.1  2004/01/29 22:13:11  pxs565
!!! Initial revision
!!!


Module Bspglocons ! Bead Spring simulation, Global constant ...
    (parameters)
  Save

  Integer, Parameter :: Ndim = 3 ! Dimension of simulation
  Integer, Parameter ::  MaxXdist = 101
  Integer, Parameter :: k4b = Selected_int_kind(9)
  Integer, Parameter :: DBprec = Selected_real_kind(8)
  Integer, Parameter :: NProps = 17
  Integer, Parameter :: MAXCHEB = 500
  Real, Parameter :: PI = 3.14159265358979323846
  Real, Parameter :: TINI = 1e-25
  Real, Parameter :: MYEPS = 1e-6
  Integer, Parameter :: HOOK = 1, FENE = 2, ILC = 3, WLC = 4, ...
     FENEFraenkel = 5
!FENE-Fraenkel Replacement
End Module Bspglocons


Module Bspglovars ! Bead Spring simulation, Global variables
  Use Bspglocons
```

```
!!$  Real Cubsoln_lu(0:1000), Gama_inc, Gama_max
!!$  Real, Allocatable ::Cubsoln_lu_2d(:,:), Gama_inc_2d(:), ...
    Gama_max_2d(:)

  Character(10) :: Prop_names(NProps)

  Character(10) , Parameter :: Correl_names(1) = (/"Sxy"/)

  Real Imploop_tol, Tstep_conv_tol
End Module Bspglovars

Module Flowvars
  Integer, Parameter :: EQ = 0, SH = 1, UA = 2, PL = 3,  UR = 4, PU ...
     = 5, PP = 6
  Integer FlowType
       ! EQ equilibrium, no flow
       ! SH Planar shear
       ! UA Uniaxial Elongational
       ! PL Planar Elongation
       ! UR Uniaxial extension followed by relaxation
       ! PU Periodic uniaxial extension
  Real gdots
end Module Flowvars


Module Bspintfacs
  Use Bspglocons

  !-----------------------------------------------------------------------c   ...

  !     Driver subroutines ...
                                                      c
  !-----------------------------------------------------------------------c

  Interface
     Subroutine Initial_position(Stype,N,L0s,Q0s,R,seed) ...
        !FENE-Fraenkel Replacement
       Use bspglocons
       Integer, Intent (in) :: Stype
```

```
      Integer(k4b), Intent (in) :: N
      Real (DBprec), intent (in) :: L0s !FENE-Fraenkel Replacement
      Real (DBprec), intent (in) :: Q0s !FENE-Fraenkel Replacement
      Integer(k4b), Intent (inout) :: seed
      Real (DBprec), Intent (out) :: R(:,:) !FENE-Fraenkel Replacement
      !Real, intent (out) :: R(Ndim,N)
    End Subroutine Initial_position
End Interface

Interface
    Subroutine GaussRand(N,GX,seed)
      Use bspglocons
      Integer(k4b), Intent (in) :: N
      Integer(k4b), Intent (inout) :: seed
      Real (DBprec), Intent (out), Dimension(N) :: GX ...
          !FENE-Fraenkel Replacement
    End Subroutine GaussRand
End Interface


Interface
    Subroutine Time_Integrate_Chain(NBeads, R_Bead, spring_type, &
         tcur, tmax, Delts, &
         Hstar, Zstar, Dstar, L0s, &
         Q0s, & !FENE-Fraenkel Replacement
         seed1, Nsamples, times, samples)

      Use bspglocons
      Integer, Intent(in) :: NBeads
      Real (DBprec), Intent (inout), Dimension(:,:) :: R_Bead ! ...
          Pos. vector of Beads !FENE-Fraenkel Replacement

      Integer, Intent (in) :: spring_type ! Spring force law type
      ! Hookean = 1, FENE = 2, ILC = 3, WLC = 4, FENE-Fraenkel = 5 ...
          !FENE-Fraenkel Replacement

      Real (DBprec), Intent (in) :: tcur,tmax,Delts  ! Integration ...
          time interval specs !FENE-Fraenkel Replacement
```

```
      Real,  Intent (in) :: Hstar         ! HI parameter (non-dim)
      Real,  Intent (in) :: Zstar,Dstar   ! EV parameters (non-dim)
      Real (DBprec),  Intent (in) :: L0s           ! finite ext., ...
         param sqrt(b) !FENE-Fraenkel Replacement
      Real (DBprec),  Intent (in) :: Q0s           ! natural ...
         length of spring !FENE-Fraenkel Replacement

      Integer (k4b), Intent (inout) :: seed1 ! seed for rnd number

      Integer, Intent(in) :: Nsamples     ! Number of sampling points
      Real (DBprec),  Intent (in), Dimension(:) :: times    ! ...
         Sampling instances !FENE-Fraenkel Replacement
      Real (DBprec), Intent (inout), Dimension(:,:) :: samples ...
         !FENE-Fraenkel Replacement
   End Subroutine Time_Integrate_Chain
End Interface

Interface
   Subroutine polish_poly_root(c,xin,atol)
     Implicit None
     ! use newton raphson to polish the root of a polynomial
     Integer, Parameter :: n=4
     Real, Intent (in) :: c(n)
     Real, Intent (inout) :: xin
     Real, Intent (in) :: atol
   End Subroutine polish_poly_root
End Interface

!-----------------------------------------------------------------------c   ...

!     Utility subroutines ...
                                                      c
!-----------------------------------------------------------------------c

Interface
   Subroutine ran_1(n, R, idum)
     Use bspglocons
     Integer(k4b), Intent(inout) ::idum
```

```
      Integer(k4b), Intent(in) :: n
      Real (DBprec), Intent(inout) :: R(n) !FENE-Fraenkel Replacement
   End Subroutine ran_1
End Interface

Interface
   Subroutine maxminev_fi(n, A, maxev, minev)
     Use bspglocons
     Integer n
     Real (DBprec) A(:,:,:,:), maxev, minev !FENE-Fraenkel ...
        Replacement
   End Subroutine maxminev_fi
End Interface

Interface
   Subroutine chbyshv (L, d_a, d_b, a)
     Use bspglocons
     Integer L
     Real (DBprec) d_a, d_b , a(0:MAXCHEB) !FENE-Fraenkel Replacement
   End Subroutine chbyshv
End Interface

Interface
   Subroutine cublu
     Use bspglocons
   End Subroutine cublu
End Interface
!-----------------------------------------------------------------------c
!     Properties estimators ...
                                                c
!-----------------------------------------------------------------------c   ...


Interface
   Subroutine chain_props(N, Rbead, F, props)
     Use bspglocons
     Implicit None
     Integer, intent (in) :: N
     Real (DBprec), Intent (in), Dimension(:,:)  :: Rbead, F ...
```

```
            !FENE-Fraenkel Replacement
          Real (DBprec), Intent (out),Dimension(:) ::props ...
            !FENE-Fraenkel Replacement
        End Subroutine chain_props
    End Interface

    Interface
        Subroutine time_correl(N, R, F, t0, t, correl)
          Use bspglocons
          Use bspglovars
          Implicit None
          Integer, intent (in) :: N
          Real (DBprec), Intent (in), Dimension(:,:)  :: R, F ...
            !FENE-Fraenkel Replacement
          Real (DBprec), Intent (out)  :: correl !FENE-Fraenkel ...
            Replacement
          Real (DBprec), Intent (in) :: t0, t !FENE-Fraenkel Replacement
        End Subroutine time_correl
    End Interface

    Interface
        Subroutine numint(f,dx,nmin,nmax,nord,sumf)
          Implicit None

          Real (Selected_real_kind(8)), Intent(in), Dimension(:) :: f ...
            !FENE-Fraenkel Replacement
          Real (Selected_real_kind(8)), Intent(in) :: dx ...
            !FENE-Fraenkel Replacement
          Real, Intent(out) :: sumf
          Integer, Intent (in) :: nmin, nmax,nord
        End Subroutine numint
    End Interface

    Interface
        Subroutine meanerr(vec,mean,err)
          Real, Intent(in) :: vec(:)
          Real, Intent(out) :: mean
          Real, Intent(out) :: err
        End Subroutine meanerr
```

```
205   End Interface

      Interface
         Function normeqpr(Stype, r, b, natscaled) !FENE-Fraenkel ...
             Replacement
           Implicit None
210        Integer, Intent (in) :: Stype
           Real, Intent (in) :: b, r, natscaled !FENE-Fraenkel Replacement
           Real  normeqpr
         End Function normeqpr
      End Interface
215

      Interface
         Function  lam1_th(hs, N)
           Real lam1_th
           Real,  Intent (in) :: hs
220        Integer,  Intent (in) :: N
         End Function lam1_th
      End Interface

      Interface
225      subroutine get_kappa(t,K)
           use bspglocons
           Implicit none
           Real (DBprec), intent (in) :: t !FENE-Fraenkel Replacement
           Real (DBprec), intent (in), dimension(:,:) :: K ...
              !FENE-Fraenkel Replacement
230      end subroutine get_kappa
      end Interface




235   End Module Bspintfacs
```

D.3 utils.f90

```
!!! Time-stamp: <utils.f90 15:32, 03 May 2018 by DP Amarasinghe>

!---------------------------------------
!  Contains general purpose utilities
!---------------------------------------

!!! $Log: utils.f90,v $
!!! Revision 1.1  2004/01/29 06:40:09  pxs565
!!! Initial revision
!!!



Subroutine Initial_position(SType,N,L0s,Q0s,R,myseed) ...
   !FENE-Fraenkel Replacement
  Use bspglocons
  Use bspglovars
  Use bspintfacs, IP_from_bspintfacs => Initial_position ...
     !FENE-Fraenkel Replacement
  !use iflport ! for portable intel intrinsic functions, here ran()
  Implicit None

  Integer, Intent (in) :: Stype
  Integer(k4b), Intent (in) :: N
  Integer(k4b), Intent (inout) :: myseed
  Real (DBprec), intent (in) :: L0s !FENE-Fraenkel Replacement
  Real (DBprec), intent (in) :: Q0s !FENE-Fraenkel Replacement
  Real (DBprec), Intent (out), Dimension(:,:) :: R !FENE-Fraenkel ...
     Replacement

  Integer, save :: varseed = 201235

  Integer mu, done
  Real, parameter :: upbound = 2.0
  Real modr,b2b(Ndim),rv,eqpr, rvchk, b, natscaled !FENE-Fraenkel ...
```

```
      Replacement


35

   Call GaussRand(Ndim*N,R,myseed)



40 R(:,1) = 0

   ! intrinsic ran() modifies the seed for every call
   ! so use a separate varible seed, so that the ran_1 sequence is
   ! not altered by this variation.
45 !varseed = seed

   b = L0s*L0s
   natscaled = Q0s/L0s !FENE-Fraenkel Replacement

50 !!$  do rv = 0,0.99,.01
   !!$     eqpr = normeqpr(Stype,rv,b);
   !!$  !THIS NEEDS TO BE MODIFIED IF YOU WISH TO USE IT FOR THE ...
      FENE-FRAENKEL !FENE-Fraenkel Replacement
   !!$     write (*,*) rv,eqpr
   !!$  end do
55

   Do mu = 2,N
      b2b = R(:,mu) ! * sqrt(sigma=1), gaussian dist betwn two adj beads



60    if (Stype .ne. HOOK) then
         done = 0;
         do while (done .ne. 1)
            if (Stype .eq. FENEFraenkel) then !FENE-Fraenkel Case
                rv = (2.0 * ran()) - 1.0 + natscaled
65          !Random number for the proportion of extension.
            !Add natscaled so that
            !rv = Q/s = Q/sqrt(b) = random + (Q0/s)
            else !Original case as in old code
                rv = ran()
```

```
            end if
            eqpr = normeqpr(Stype,rv,b,natscaled); !FENE-Fraenkel ...
               Replacement

            ! use another uniform random number to decide acceptance
            ! ie, accept only eqpr fraction at this r
            ! If variance reduction is not being used then simply ...
               use ran() rather than ran(varseed)
            rvchk = ran()
            !write (*,*) 'rv = ', rv
            !write(*,*) 'eqpr = ', eqpr
            !write(*,*) 'rvchk = ', rvchk
            if (rvchk .le. eqpr/upbound) done = 1
         end do
         modr = Sqrt(Sum(b2b*b2b))
         b2b = b2b/modr * (rv * L0s)
         !b2b/modr gives a random direction for a unit vector ...
            !FENE-Fraenkel Replacement
         !(rv * L0s) = Q !FENE-Fraenkel Replacement
      end if

      R(:,mu) = R(:,mu-1) + b2b

      ! in the case of a finitely extensible spring, a gaussian ...
         distribution
      ! can lead to incorrect forces some exceptional b2b vectors, ...
         we limit
      ! this by altering the distance when preseving the random ...
         direction.
      ! and an additional random factor between 0 and 1
      ! A good guess of the factor could be obtained more rigorously.
  End Do
End Subroutine Initial_position

Subroutine GaussRand(N,GX,seed)
!!! Returns a normalised gaussian random variable with mean zero
!!! and variance 1: f(r) = exp(-r^2/2) / sqrt(2*Pi)
!!! To obtain a dist with a given sigma use:  sqrt(sigma) * Gx
```

```
  Use bspglocons
  Use bspintfacs, GR_from_bspintfacs => GaussRand !FENE-Fraenkel ...
     Replacement
  Implicit None

  Integer(k4b), Intent (in) :: N
  Integer(k4b), Intent (inout) :: seed
  Real (DBprec), Intent (out), Dimension(N) :: GX !FENE-Fraenkel ...
     Replacement

  Real (DBprec) rnd(2) !FENE-Fraenkel Replacement
  Real x,y, rsq, fac !FENE-Fraenkel Replacement
  Integer i,n2
  Real :: rtemp(N+1)


  Do i = 1,N,2
     rsq = 0.0
     Do While (rsq .Ge. 1.0  .Or.  rsq .Eq. 0.0)
        Call ran_1(2,rnd,seed)
        x = 2*rnd(1) -1
        y = 2*rnd(2) -1
        rsq = x*x + y*y
     End Do
     fac = Sqrt(-2*Log(rsq)/rsq);
     rtemp(i) = x*fac
     rtemp(i+1) = y*fac
  End Do


  GX = rtemp(1:N)

End Subroutine GaussRand



Subroutine ran_1(n, X, idum)
  Use bspglocons
  Implicit None
  Integer(k4b), Intent(inout) ::idum
```

```
140   Integer(k4b), Intent(in) :: n
      Real (DBprec), Intent(inout) :: X(n) !FENE-Fraenkel Replacement
      !-------------------------------------------------------------------------c
      !     Random number generator based on procedure given in ...
                      c
      !     Numerical Recipes in Fortran 90, Chapter B7, pp. 1141-1143 ...
              c
145   !     This generator is supposed to have a period of about ...
         3.1E18.    c
      ! ...
                                                                              ...
         c
      !     The calling sequence is the same as that of RANL, ...
                        c
      !     the BLAS random number generator. The arguments are: ...
                      c
      !     n - gives the dimension of vector R ...
                                      c
150   !     X - 1D array of dimension n; on exit ...
                                    c
      !         contains n unifromly distributed random numbers in ...
         (0,1)    c
      !     idum |- seeds which are updated on exit ...
                                 c
      !-------------------------------------------------------------------------c

155   Integer(k4b),Parameter :: IA=16807,IM=2147483647,IQ=127773,IR=2836
      Real, Save :: am
      Integer(k4b), Save :: ix=1, iy = -1, k
      Integer(k4b) count

160   If ((idum.Le.0) .Or. (iy.Lt.0)) Then
         am = Nearest(1.0,-1.0)/IM
         iy = Ior(Ieor(888889999,Abs(idum)),1)
         ix = Ieor(777755555,Abs(idum))
         idum = Abs(idum) + 1
165   End If

      Do count = 1,n
```

```
      ix = Ieor(ix,Ishft(ix,13))
      ix = Ieor(ix, Ishft(ix,-17))
170   ix = Ieor(ix,Ishft(ix, 5))
      k = iy/IQ
      iy = IA*(iy-k*IQ)-IR*k
      If (iy.Lt.0) iy = iy + IM
      X(count) = am*Ior(Iand(IM,Ieor(ix,iy)),1)
175   End Do

End Subroutine ran_1


180 Subroutine chbyshv (L, d_a, d_b, a)
    Use bspglocons
    Implicit None
    Integer L
    Real (DBprec) d_a, d_b, a(0:MAXCHEB) !FENE-Fraenkel Replacement
185 !---------------------------------------------------------------------c
    !     This routine calculates the Chebyshev coefficients for the ...
            c
    !     Chebyshev polynomial approximation of a square root ...
       function.   c
    !     The routine computes L+1 coefficients, which are returned ...
       in    c
    !     in the one-dimensional array a, for the approximation of ...
       the    c
190 !     square root of any variable that lies in the interval ...
                c
    !     (d_a, d_b). ...
                                                              c
    !---------------------------------------------------------------------c


195 Integer j, k
    Real xks(0:L)


    !     Calculate the shift factors
200 ! d_a = 2/(lmax-lmin)
```

```
!      -d_b/d_a = (lmax+lmin)/2

!     Calculate the collocation points
Do k = 0,L
   xks(k) = Cos(PI*(k+0.5)/(L+1))/d_a -d_b/d_a
End Do

!     Calculate the Chebyshev coefficients
a = 0.0
Do j = 0,L
   Do k = 0,L
      a(j) = a(j)+(xks(k)**0.5)*Cos(j*(k+0.5)*PI/(L+1))
   End Do
   a(j) = 2.0/(L+1)*a(j)
End Do
a(0) = a(0)/2

End Subroutine chbyshv



Subroutine maxminev_fi(n, A, maxev, minev)
  Use bspglocons
  Implicit None
  Integer n
  Real (DBprec) A(:,:,:,:), maxev, minev !FENE-Fraenkel Replacement
  !-----------------------------------------------------------------------c
  !     This routine approximates the maximum and minimum eigen ...
     values  c
  !     of a symmetric matrix nxn matrix A using Fixman's ...
     suggestion.   c
  !-----------------------------------------------------------------------c
  Integer lda, incx, incy, i
  Real (DBprec) F(n), G(n), alpha, beta, ddot !FENE-Fraenkel ...
     Replacement

  F = 1.D0 !FENE-Fraenkel Replacement
```

```
  G = 0.D0 !FENE-Fraenkel Replacement

  lda = n
  alpha = 1.D0 !FENE-Fraenkel Replacement
  beta = 0.D0 !FENE-Fraenkel Replacement
  incx = 1
  incy = 1

  Call dsymv('U',n,alpha,A,lda,F,incx,beta,G,incy) !FENE-Fraenkel ...
     Replacement
  maxev = ddot(n, F, 1, G, 1) !FENE-Fraenkel Replacement
  maxev = 2.D0*maxev/n !FENE-Fraenkel Replacement

  !Forall (i = 1:n) F(i) = (-1.0)**i
  Do i = 1,n,2
     F(i)   = -1.D0 !FENE-Fraenkel Replacement
     F(i+1) =  1.D0 !FENE-Fraenkel Replacement
  End Do


  Call dsymv('U',n,alpha,A,lda,F,incx,beta,G,incy) !FENE-Fraenkel ...
     Replacement
  minev = ddot(n, F, 1, G, 1) !FENE-Fraenkel Replacement
  minev = minev/2.D0/n !FENE-Fraenkel Replacement
  !        write (*,*) maxev, minev

End Subroutine maxminev_fi




Subroutine polish_poly_root(c,xin,atol)
  Implicit None
  ! use newton raphson to polish the root of a polynomial
  Integer, Parameter :: n=4 ! presently only for cubic
  Real, Intent (in) :: c(n)
  Real, Intent (inout) :: xin
  Real, Intent (in) :: atol
```

```
    Real ::  p, p1,x0,x

    Integer i,iter
    Integer, Parameter :: IMAX = 30

    x = xin

    ! algo from NR: techniques for polishing, roots of polynomials
    Do iter=1,IMAX
       x0 = x
       p =  c(n)*x + c(n-1)
       p1 = c(n)
       Do i=n-2,1,-1
          p1 = p + p1*x
          p  = c(i) + p*x
       End Do

       !if (abs(p1) < atol) then ! f' = 0 is not solvable in newt-raph
       If (Abs(p1) .Eq. 0.0) Then ! f' = 0 is not solvable in newt-raph
          !! for WLC, in this case f'' also = 0, therefore
          p1 = 6 * c(4) ! f'''
          p1 = 6*p/p1
          ! note sign(a,b) returns sgn(b) * mod(a)
          x = x - Sign(1.0,p1) * Abs(p1)**(1./3.)
          !! we omit considering cases for ILC and FENE, as they
          !! donot seem to have this singularity
       Else
          x = x - p/p1
       End If

       If (Abs(p) .Le. atol) Exit
    End Do
    if (iter>IMAX) then
       !write (5,*) 'poly-root: loop exceeded'
    end if


    xin = x
  End Subroutine polish_poly_root
```

```
315 Subroutine numint(f,dx,nmin,nmax,nord,sumf)
    Implicit None

    Real (Selected_real_kind(8)), Intent(in), Dimension(:) :: f ...
       !FENE-Fraenkel Replacement
    Real (Selected_real_kind(8)), Intent(in) :: dx !FENE-Fraenkel ...
       Replacement
320 Real, Intent(out) :: sumf
    Integer, Intent (in) :: nmin, nmax,nord



325 Integer, Parameter :: stdout=5, stderr=6
    Integer nbound,nint,j

    nbound = Ubound(f,1)
    nint = nmax-nmin
330

    !write (*,*) 'nbound = ', nbound


    If (nint > nbound .Or. nmin < 1 .Or. nmax > nbound ) Then
335    Write(stderr,*) 'Array out of bounds: ', nmin,nmax,nbound
       Return
    End If

    sumf = 0.
340

    Select Case (nord)
    Case (1) ! trapezoidal rule
       sumf = 0.5 * (f(nmax) + f(nmin))

345 Case(2)
       sumf = 5./12 * (f(nmin) + f(nmax)) + &
            13./12 *(f(nmin+1) + f(nmax-1))
    Case (3)
       sumf = 3./8 * (f(nmin) + f(nmax)) + &
```

```
350           7./6 *(f(nmin+1) + f(nmax-1)) + &
              23./24 *(f(nmin+2) + f(nmax-2))
    End Select

    Do j = nmin+nord, nmax-nord
355       sumf = sumf + f(j)
    End Do
    sumf =  dx*sumf

  End Subroutine numint

360
  subroutine meanerr(vec,mean,err)
    real, intent(in) :: vec(:)
    real, intent(out) :: mean
    real, intent(out) :: err ! standard error of mean
365
    integer n

    n = ubound(vec,1)
    mean = sum(vec)/n
370
    ! though the following is correct,
    ! err = sqrt((sum(vec*vec) - mean*mean*n)/(n-1))
    ! it does not always numerically evaluate to a quantity > 0
    ! for very small errors, such as in the case of diffusivity.
375 ! so use the sure shot formula.
    err = 2*sqrt(sum((vec-mean)*(vec-mean))/(n-1)/n)
    ! the standard error of mean is approximately in the 95% confidence
    ! interval, giving the factor 2 (from the t-distribution)


380
  end subroutine meanerr

  function normeqpr(Stype, r, b, natscal) !FENE-Fraenkel Replacement
    Use bspglocons !FENE-Fraenkel Replacement
385 implicit none
    Integer, intent (in) :: Stype
    Real, intent (in) :: b, r, natscal !FENE-Fraenkel Replacement
    Real normeqpr
```

```
390   Real expphi
      external expphi

      Integer, save :: norm_computed = 0
      Real   , save :: norm_b, bsav
395

      Integer, parameter :: Ln = 21 ! 21 always see below for X,W data
      Real (8) X(Ln), W(Ln), scratch(Ln)
      Real (8) endpts(2)

400   Integer n
      Real xfact, rg, fb, M

      if (norm_computed .eq. 0 .and. bsav .ne. b) then

405      !! generate the gauss abscissa and weights for legendre
         !call gaussq(1, Ln, 0, 0, 0, endpts, scratch, X, W) ;

         !! generated from c code
         X(01) = -0.993752170620389;   W(01) = 0.016017228257774
410      X(02) = -0.967226838566306;   W(02) = 0.03695378977085309
         X(03) = -0.920099334150401;   W(03) = 0.05713442542685689
         X(04) = -0.853363364583318;   W(04) = 0.0761001136283793
         X(05) = -0.768439963475678;   W(05) = 0.09344442345603385
         X(06) = -0.667138804197413;   W(06) = 0.1087972991671478
415      X(07) = -0.55161883588722;   W(07) = 0.1218314160537286
         X(08) = -0.424342120207439;   W(08) = 0.1322689386333376
         X(09) = -0.288021316802401;   W(09) = 0.1398873947910733
         X(10) = -0.145561854160895;   W(10) = 0.14452440398997
         X(11) = -2.4782829604619e-16;   W(11) = 0.1460811336496907
420      X(12) = 0.145561854160895;   W(12) = 0.1445244039899697
         X(13) = 0.288021316802401;   W(13) = 0.1398873947910732
         X(14) = 0.424342120207439;   W(14) = 0.1322689386333371
         X(15) = 0.55161883588722;   W(15) = 0.1218314160537288
         X(16) = 0.667138804197413;   W(16) = 0.1087972991671492
425      X(17) = 0.768439963475678;   W(17) = 0.09344442345603389
         X(18) = 0.853363364583317;   W(18) = 0.07610011362837897
         X(19) = 0.920099334150401;   W(19) = 0.05713442542685797
```

```
      X(20) = 0.967226838566306;   W(20) = 0.03695378977085323
      X(21) = 0.993752170620389;   W(21) = 0.01601722825777395

      M = 18  ! some large number, obtained by trial and error

      ! limit the upper bound of M
      if (M > b/2) M = b/2

      xfact = sqrt(2*M/b)

      norm_b = 0
      fb = 0

      if (Stype .eq. FENEFraenkel) then !FENE-Fraenkel Case
         do n=1,Ln
             rg = (X(n) * xfact) + natscal
             norm_b = norm_b +  W(n) * expphi(Stype,rg,b,natscal) * ...
                 rg * rg
             fb = fb + W(n) * expphi(Stype,rg,b,natscal) * rg**4
         end do
      else !Original case as in old code
         do n=1,Ln
             rg = (X(n) + 1)/2 * xfact
             norm_b = norm_b +  W(n) * expphi(Stype,rg,b,natscal) * ...
                 rg * rg
             fb = fb +  W(n) * expphi(Stype,rg,b,natscal) * rg**4
         end do
      end if !FENE-Fraenkel Replacement

      !write (*,*) '# b = ', b
      !write (*,*) '# computed normb = ', norm_b
      !write (*,*) '# f(b) = ', b*fb/norm_b

      bsav = b
      norm_computed = 1
   end if

   normeqpr = r*r*expphi(Stype,r,b,natscal)/norm_b !FENE-Fraenkel ...
       Replacement
```

```
465 end function normeqpr


    function expphi(Stype, r, b, natlensc) !FENE-Fraenkel Replacement
    !!! return the distribution function (without normalisation)
470   use bspglocons
      Implicit none
      Integer, intent (in) :: Stype
      Real, intent (in) :: r, b, natlensc !FENE-Fraenkel Replacement
      Real  expphi
475

      select case (Stype)
    !CAREFUL - IN ALL CASES, THESE ARE -1 TIMES THE ACTUAL EXPRESSION ...
       DERIVED !FENE-Fraenkel Replacement
      case (HOOK)
         write (*,*) 'Hookean case called in expb: Check the code'
480   case (FENE)
         expphi = (1-r*r)**(b/2)
      case (WLC)
         expphi = exp( -b/6.0 * ( 2*r*r + 1/(1-r) -r -1 ))
      case (FENEFraenkel)
485      expphi = (1-((r-natlensc)*(r-natlensc)))**(b/2); ...
            !FENE-Fraenkel Replacement
     end select
    end function expphi


490 function  lam1_th(hs, N)
      use bspglocons
      implicit none
      Real lam1_th
      Real,  intent (in) :: hs
495   Integer,  intent (in) :: N


      real s,b,aj

500
```

```
    b   = 1 - 1.66*hs**0.78;
    s   =   - 1.40*hs**0.78;

    aj = 4*sin(Pi/2./N)*sin(Pi/2./N);
    lam1_th = 2./( aj * b / N**s);

  end function lam1_th

  subroutine get_kappa(t,K)
    use bspglocons
    use Flowvars
    Implicit none
    Real (DBprec), intent (in) :: t !FENE-Fraenkel Replacement
    Real (DBprec), intent (out), dimension(:,:) :: K !FENE-Fraenkel ...
       Replacement


    Real omgs,var,T0



    K  = 0

    Select Case (FlowType)
    case (EQ) ! Equilibrium
       K  = 0
    case (UA) ! uniaxial extension
       K(1,1) = 1 !FENE-Fraenkel Replacement
       K(2,2) = -0.5 !FENE-Fraenkel Replacement
       K(3,3) = -0.5 !FENE-Fraenkel Replacement
    case (PL) ! planar extension
       K(1,1) = 1
       K(2,2) = -1
    case (SH) ! planar shear
       K(1,2) = 1
    case (UR) ! uniaxial extension followed by relaxation
       if (t .le. 5./gdots) then
          K(1,1) = 1
          K(2,2) = -0.5
          K(3,3) = -0.5
```

```
        end if
  case (PU) ! periodic uniaxial extension

        T0 = 10.0 ! time period of oscilation in strain units

        omgs = 2*PI/T0 * gdots
        var = sin(omgs*t)
        K(1,1) = 1
        K(2,2) = -0.5
        K(3,3) = -0.5
        K = K * var

  case (PP) ! periodic planar extension

        T0 = 10.0 ! time period of oscilation in strain units

        omgs = 2*PI/T0 * gdots
        var = sin(omgs*t)
        K(1,1) = 1
        K(2,2) = -1
        K = K * var
  end Select

  K = K * gdots

end subroutine get_kappa


!*****************************************************************!***********
! This module contains utilities to lock files to prevent other ...
   programs
! to open and write data into it.  This is done by creating another ...
   file in
! the directory called lock-<file>, where <file> is the file to be ...
   locked.
! The module contains routines for locking, unlocking (removing the ...
   lock-<file>
! and for returning the status of a lock through islocked()
! The unlocking is done by 'deleting' the lock-<file>
```

```
!*****************************************************************!***********

Module flock_utils
Implicit none

!--string prefix for the lock file
Character(5), parameter :: prefx = 'lock-'
!--a constant number to be added to the unit of the file to be locked
Integer, parameter :: added = 100

Contains

  Subroutine islocked(file,lstat)

    Character(*), intent(in) :: file
    Logical, intent(out) :: lstat

    Character(40) :: lfile

    lfile = prefx // file

  Inquire (file=lfile, exist=lstat)

  end Subroutine islocked

  Subroutine lockfile(unit,file)
    Integer, intent(in) :: unit
    Character(*), intent(in) :: file

    Integer lunit
    Character(40) :: lfile

    lfile = prefx // file

    lunit = unit + added
    open(unit=lunit,file=lfile)

  end Subroutine lockfile
```

```
  Subroutine unlockfile(unit,file)

    Integer, intent(in) :: unit
    Character(*), intent(in) :: file

    Integer lunit
    Character(40) :: lfile

    lfile = prefx // file

    lunit = unit + added
    close(unit=lunit,status='delete')


  end Subroutine unlockfile
end Module flock_utils
```

D.4 gsipc.f90

```
!!! Time-stamp: <gsipc.f90 15:32, 03 May 2018 by DP Amarasinghe>

!-----------------------------------------------
!   Bead Spring Configuration Space Utilities
!-----------------------------------------------

!!! $Log: gsipc.f90,v $
!!! Revision 1.3  2004/02/09 05:38:36  pxs565
!!! pcerr made relative
!!!
!!! Revision 1.2  2004/02/03 02:02:35  pxs565
!!! cofm initialised correctly
!!!
!!! Revision 1.1  2004/01/29 22:11:08  pxs565
!!! Initial revision
!!!


Module csputls
!!! This module contains subroutines used to manipulate the
!!! configuration variable, R and its derivatives
  Use bspglocons

Contains

  Subroutine b2bvector_sym_up(N,R,b2b)
    Implicit None
    ! Calling arguments
    Integer, Intent(in)  :: N
    Real (DBprec), Intent(in)  :: R(:,:) !FENE-Fraenkel Replacement
    Real (DBprec), Intent(out) :: b2b(:,:,:) !FENE-Fraenkel Replacement

!!!-------------------------------------------------------|
!!! This routine returns the vector one bead to another bead
!!! as a symmetric matrix but only the upper elements filled
```

```
!!!-----------------------------------------------------------|

    Integer mu,nu

    ! according to /usr/bin/prof, this takes max time. so ...
       commenting out
    ! since only upper diagonal is anyway reqd
    ! b2b = 0

    ! note only upper half (nu > mu) of the matrix is filled
    ! and the convention is R_12 = R_2 - R_1
    ! where R_12 is the vector from 1 to 2
    Do nu = 2,N
       Do mu = 1,nu-1
          b2b(:,mu,nu) = R(:,nu) - R(:,mu)
       End Do
    End Do
  End Subroutine b2bvector_sym_up

  Subroutine modr_sym_up(N,b2bvec,deltaR)
    Implicit None
    ! calling arguments
    Integer, Intent (in)  :: N
    Real (DBprec), Intent (in)  :: b2bvec(:,:,:) !FENE-Fraenkel ...
       Replacement
    Real (DBprec), Intent (out) :: deltaR(:,:) !FENE-Fraenkel ...
       Replacement
!!!-----------------------------------------------------------|
!!! This subroutine returns the magnitude of each of the bead
!!! to bead vector
!!!-----------------------------------------------------------|

    Integer mu,nu,i
    Real r12(Ndim),modr



    deltaR = 0

    ! note that we cant use a forall since dot_product is a
```

```
      ! transformational function
      Do nu = 2,N
         Do mu = 1,nu-1
            r12 = b2bvec(:,mu,nu)
            modr = 0.0
            Do i=1,Ndim
               modr = modr + r12(i) * r12(i)
            End Do
            If (modr < 1.0e-12) modr = 1.0e-12   !so that initial dr ...
               is != 0
            deltaR(mu,nu) = Sqrt(modr)
         End Do
      End Do
   End Subroutine modr_sym_up


   Subroutine tensor_prod_of_vec(alpha,vec,tensor)
      Implicit None
      Real (DBprec), Intent (in) :: alpha !FENE-Fraenkel Replacement
      Real (DBprec), Intent (in) :: vec(:) !FENE-Fraenkel Replacement
      Real (DBprec), Intent (out) ::  tensor(:,:) !FENE-Fraenkel ...
         Replacement
      Integer i,j
      Real temp

      Do j = 1,Ndim
         Do i = 1,j
            temp = alpha * vec(i) * vec(j)
            tensor(i,j) = temp
            tensor(j,i) = temp
         End Do
      End Do
   End Subroutine tensor_prod_of_vec
End Module csputls


Module BSpModel
!!! This module contains the constants and functions which are specific
!!! to the model of spring and EV.
```

```
  Use bspglocons
  Implicit None

  ! These are paramters that will be used  by the following procedures
  Real (DBprec) sqrtb, natlength !FENE-Fraenkel Replacement

  ! paramters for EV
  Real zsbyds5,pt5bydssq
  ! parameters for LJ
  !Real ktbyepsilon,sigmabylk

Contains

  Subroutine force_sans_hookean(sptype,r,ff) !FENE-Fraenkel Replacement
    Integer, Intent (in) :: sptype

    Real (DBprec), Intent (out) :: ff

    Real (DBprec) r
    Real (DBprec) scalednat !FENE-Fraenkel Replacement

    !r = |Q| / sqrtb  !COMMENT BY PRAVEEN
    scalednat = natlength/sqrtb ! = |Q0| / sqrtb  !FENE-Fraenkel ...
       Replacement
    ! SCALED (by root b) natural spring length for the ...
       FENE-Fraenkel case.

    ! compute the force factor, ff
    Select Case (sptype)
    Case (HOOK) ! Hookean
       ff = 1.0

    Case (FENE) ! Warner Spring
       ff = 1./(1 - r*r)

    Case (ILC)  ! Inverse Langevin, Pade approx
       ff = (3-r*r)/(1 - r*r)/3.

    Case (WLC) ! Worm-like Chain, Marko-Siggia interpolation
```

```
         ff = 1./(6*r) * ( 4*r+ 1./(1-r)/(1-r) - 1)

       Case (FENEFraenkel) ! FENE-Fraenkel Spring
         ff = (1.-(scalednat/r))/(1.-((r-scalednat)*(r-scalednat)))
150    !FENE-Fraenkel Replacement

       End Select
     End Subroutine force_sans_hookean

155  Subroutine Spring_force(sptype,N,b2bvec,dr,Fs)
       Integer, Intent (in) :: sptype
       Integer, Intent (in) :: N
       Real (DBprec), Intent (in)  :: b2bvec(:,:,:)! Ndim x Nbeads x ...
          Nbeads !FENE-Fraenkel Replacement
       Real (DBprec), Intent (in)  :: dr(:,:)      ! Nbeads x Nbeads ...
          !FENE-Fraenkel Replacement
160    Real (DBprec), Intent (out) :: Fs(:,:)      ! Ndim x Nbeads ...
          !FENE-Fraenkel Replacement
   !!! Purpose:
       ! computes the net spring force on a bead due to
       ! connector springs of neighbouring beads

165    Integer nu
       Real (DBprec) r ! = |Q| !SCALING BY sqrtb HAPPENS BELOW! ...
          !FENE-Fraenkel Replacement
       Real (DBprec) Fcnu(Ndim), Fcnu1(Ndim), nonhook

       Fs  = 0.0
170

       Fcnu1 = 0.0 ! Fc of nu-1

       Do nu = 1,N-1 ! over each of the connector

175       r = dr(nu,nu+1)/sqrtb ! only upper diagonal
          !SCALING OF SPRING LENGTH BY sqrtb happens here! ...
             !FENE-Fraenkel Replacement
          If (Abs(r - 1.0) .Lt. MYEPS) r = 1 - MYEPS

          Call force_sans_hookean(sptype,r,nonhook)
```

```
180
          ! connector force from 1 to 2 (nu to nu+1)
          Fcnu = b2bvec(:,nu,nu+1) * nonhook

          ! total spring force on a bead
185       Fs(:,nu) = Fcnu - Fcnu1
          Fcnu1 = Fcnu ! save for the next bead
        End Do
        !write (*,*) r
        ! and the last bead
190     Fs(:,N) = - Fcnu1

      End Subroutine Spring_force

      !FENE-Fraenkel Replacement BELOW
195   Subroutine solve_implicit_r(sptype,dtby4,gama,natscl,r,ff,r_old)
        Integer, Intent (in) :: sptype
        Real (DBprec), Intent (in) :: dtby4, gama, natscl
        Real (DBprec), Intent (out) :: r
        Real (DBprec), Intent (out) :: ff
200
        Complex (DBprec) :: Solutions(3)
        Complex (DBprec) :: zA, zB, zC
        Real (DBprec) :: rA, rB, rC
        Real (DBprec), Intent (in) :: r_old
205     Real (DBprec) :: dif

        !! Purpose
        ! solves the equation r ( 1 + dt/4 ff ) == gama
        ! where ff depends on spring force law F = H Q ff,
210     ! for r and returns, r and ff(r)

        Real (DBprec) :: coeff(4)
        Real (DBprec) :: denom

215     ! set up the polynomial equation (cubic), and the guess for
        ! gama >> 1, obtained by the asymptotic behaviour

        coeff(4) = 1.D0
```

```
220     Select Case (sptype)
        Case (HOOK) ! Hookean
           r = gama/(1.D0+dtby4)
           ff = 1.D0
           Return
225
        Case (FENE) ! FENE
           coeff(1) = gama
           coeff(2) = -(1.D0 + dtby4)
           coeff(3) = -gama
230        r = 1.D0 - dtby4/2.D0/gama

        Case (ILC) ! ILC
           denom = 3.D0 + dtby4
           coeff(1) = 3.D0 * gama / denom
235        coeff(2) = -(3.D0 + 3.D0* dtby4) / denom
           coeff(3) = -3.D0 * gama / denom
           r = 1.D0 - dtby4/3.D0/gama

        Case (WLC) ! WLC
240        denom = 2.D0*(1.5D0 + dtby4)
           coeff(1) = -3.D0 *gama/denom
           coeff(2) =  3.D0 * (1.D0 + dtby4 + 2.D0*gama) / denom
           coeff(3) = -1.5D0 * (4.D0 + 3.D0*dtby4 + 2.D0*gama) / denom
           r = 1.D0 - Sqrt(dtby4/6.D0/gama)
245
        Case (FENEFraenkel) ! FENE-Fraenkel
           coeff(1) = (dtby4*natscl) + gama - (gama*natscl*natscl)
           coeff(2) = (-1.D0) + (natscl*natscl) - dtby4 + (2.D0 * gama ...
              * natscl)
           coeff(3) = (-2.D0 * natscl) - gama
250        Solutions = CubicSolver(coeff)
           zA = Solutions(1)
           zB = Solutions(2)
           zC = Solutions(3)
        !(FENE Fraenkel uses CubicSolver AND double precision numbers used)
255
        End Select
```

```
      !write (*,*) "Gamma value calculated is ", gama
      !write (*,*) "Polynomial is x^3 + ", coeff(3), "x^2 + ", ...
         coeff(2), "x + ", coeff(1)
      !write (*,*) "with dtby4 equal to", dtby4
      !write (*,*) "First guess for r is ", r
      !write (*,*) "The solver's numbers are ", zA, zB, zC

      ! the Hookean guess is same for all the force laws
      If (gama < 1.0) Then
         r = gama/(1.+dtby4)
      End If

      ! all the common forces laws yeild a cubic in the implicit form
      ! polish the guessed root by a Newt-Raph for polynomials

      ! for WLC, the newt raph, does not converge to the correct root,
      ! for gama > 100 , therefore simply ignore polishing
      If (sptype .Eq. WLC .And. gama > 100) Then
         r = r
      Else if (sptype .eq. FENEFraenkel) Then
         !Call polish_poly_root(coeff,rA,1e-6)
         !Call polish_poly_root(coeff,rB,1e-6)
         !Call polish_poly_root(coeff,rC,1e-6)
         !write (*,*) "Roots after polishing are ", rA, rB, rC

         r = 0.D0
         if (abs(imagpart(zA)) .lt. 1d-6) then
            rA = realpart(zA)
         else
            rA = 3.D0*natscl + 1.D0
         end if
         if (abs(imagpart(zB)) .lt. 1d-6) then
            rB = realpart(zB)
         else
            rB = 3.D0*natscl + 1.D0
         end if
         if (abs(imagpart(zC)) .lt. 1d-6) then
            rC = realpart(zC)
```

```
295         else
               rC = 3.D0*natscl + 1.D0
            end if
            !write (*,*) "The solver's real solutions are ", rA, rB, rC
            if ((rA .le. (natscl - 1.D0)) .or. (rA .ge. (natscl + ...
               1.D0))) then
300            rA = 3.D0*natscl + 1.D0
            end if
            if ((rB .le. (natscl - 1.D0)) .or. (rB .ge. (natscl + ...
               1.D0))) then
               rB = 3.D0*natscl + 1.D0
            end if
305         if ((rC .le. (natscl - 1.D0)) .or. (rC .ge. (natscl + ...
               1.D0))) then
               rC = 3.D0*natscl + 1.D0
            end if

            dif = min(abs(rA-r_old),abs(rB-r_old),abs(rC-r_old))
310         If (dif .eq. abs(rA-r_old)) then
               r = rA
            Else If (dif .eq. abs(rB-r_old)) then
               r = rB
            Else If (dif .eq. abs(rC-r_old)) then
315            r = rC
            End If
            if ((r .le. (natscl - 1.D0)) .or. (r .ge. (natscl + 1.D0))) then
               r = 1.D0*natscl + 1.D0
            end if
320      Else
            Call polish_poly_root(coeff,r,1.D-6)
         End If
         !write (*,*) "Final guess for r is ", r
         Call force_sans_hookean(sptype,r,ff)
325      !write (*,*) "Force_sans_hookean output is ", ff
      End Subroutine solve_implicit_r
      !FENE-Fraenkel Replacement ABOVE

      !FENE-Fraenkel Replacement INSERTION BELOW
330   Function CubicSolver(coeff)
```

```
!This cubic solver is based on that found in Section 5.6 of
!Numerical Recipes in Fortran: The Art of Scientific Computing,
!2nd edition (1992), Press et al.
!
!Alternative available at:
!http://blogs.warwick.ac.uk/bibojiang/entry/cubic_funtion_computational/
!http://www-old.me.gatech.edu/energy/andy_phd/appA.htm

  Real (DBprec), Intent (in) :: coeff(4)
  Real (DBprec) :: c(4)
  Complex (DBprec) :: CubicSolver(3)
  Complex (DBprec) :: x1, x2, x3

  Real (DBprec), Parameter :: pi = 3.14159265358979323846D0

  Real (DBprec) :: Q, R, diff

  Real (DBprec) :: theta, sqrtQ

  Real (DBprec) :: A, B
  Real (DBprec) :: sqrt3by2
  Real (DBprec) :: impart
  Real (DBprec) :: repart1, repart2

  c(1) = coeff(1)/coeff(4)
  c(2) = coeff(2)/coeff(4)
  c(3) = coeff(3)/coeff(4)
  c(4) = 1.D0

  Q = ((c(3)*c(3)) - (3.D0*c(2))) / 9.D0
  !write(*,*) "Q is", Q
  R = ((2.D0*c(3)*c(3)*c(3)) - (9.D0*c(2)*c(3)) + (27.D0*c(1))) / ...
     54.D0
  !write(*,*) "R is", R
  diff = (R*R) - (Q*Q*Q)

  if (diff .le. 0.) then
     theta = acos(R/(sqrt(Q*Q*Q)))
     !write(*,*) "R/sqrt(Q3) is", R/(sqrt(Q*Q*Q))
```

```
        !write(*,*) "theta is", theta
        sqrtQ = sqrt(Q)
        x1 = complex((-2.D0 * sqrtQ * cos(theta/3.D0) - ...
            (c(3)/3.D0)),0.D0)
        x2 = complex((-2.D0 * sqrtQ * cos((theta + (2.D0*pi))/3.D0) ...
            - (c(3)/3.D0)),0.D0)
        x3 = complex((-2.D0 * sqrtQ * cos((theta + (4.D0*pi))/3.D0) ...
            - (c(3)/3.D0)),0.D0)
    else
        !write(*,*) "Imaginary roots present!"
        A = sign(1.D0,R)*(((abs(R)) + sqrt(diff))**(1.D0/3.D0))
        B = 0.D0
        if (A .ne. 0.D0) B = Q/A
        sqrt3by2 = sqrt(3.D0)/2.D0
        repart1 = (A + B) - (c(3)/3.D0)
        repart2 = ((-0.5D0)*(A + B)) - (c(3)/3.D0)
        impart = sqrt3by2 * (A - B)
        x1 = complex(repart1, 0.D0)
        x2 = complex(repart2, impart)
        x3 = complex(repart2, (-1.D0*impart))
    end if

    CubicSolver(1) = x1
    !write(*,*) "x1 is ", x1
    CubicSolver(2) = x2
    !write(*,*) "x2 is ", x2
    CubicSolver(3) = x3
    !write(*,*) "x3 is ", x3

    !VERIFICATION
    !write(*,*) "Cube root of 27 is", 27.**(1./3.)
    !write(*,*) "Plugging in the solver values gives ", ...
        (coeff(4)*x1*x1*x1) + (coeff(3)*x1*x1) &
    !+ (coeff(2)*x1) + (coeff(1)), (coeff(4)*x2*x2*x2) + ...
        (coeff(3)*x2*x2) + (coeff(2)*x2) + (coeff(1)), &
    !(coeff(4)*x3*x3*x3) + (coeff(3)*x3*x3) + (coeff(2)*x3) + ...
        (coeff(1))

End Function CubicSolver
```

```
!FENE-Fraenkel Replacement INSERTION ABOVE

Subroutine Excluded_volume_force(N,b2bvec,dr,Fev)
  Use bspglocons

!!! This routine returns excluded volume force
  Integer, intent (in) :: N
  Real (DBprec), Intent (in)  :: b2bvec(:,:,:)! Ndim x Nbeads x ...
     Nbeads !FENE-Fraenkel Replacement
  Real (DBprec), Intent (in)  :: dr(:,:)      ! Nbeads x Nbeads ...
     !FENE-Fraenkel Replacement
  Real (DBprec), Intent (out) :: Fev(:,:)     ! Ndim x Nbeads ...
     !FENE-Fraenkel Replacement


  Integer nu,mu
  Real Fpair12(Ndim)

  Fev  = 0.0

! Narrow Gaussian
  If (zsbyds5.Gt.0.0) Then
     ! caution donot use forall here, i means F12 will b ...
        evaluated for
     ! all the values of mu,nu and then assigned, which is not ...
        what we
     ! want.  The rule to use forall statements is that lhs must be
     ! unique for-all the values of loop index, and RHS must not ...
        depend
     ! on any other index of lhs, other than the lhs itself
     ! ie, we can have  foo(mu) = foo(mu) + bar(nu)
     ! but not  foo(mu) = foo(mu+3) + bar(nu)
     Do nu = 2,N
        Do mu  = 1,nu-1
           ! force between the pair mu,nu, since the EV force is
           ! repulsive, we take it to be in the opposite direction
           ! of the connector vector mu -> nu
           ! convention followed: force is positive for attraction
           Fpair12 = - b2bvec(:,mu,nu) &
```

```
                    * zsbyds5*Exp(-dr(mu,nu)*dr(mu,nu) * pt5bydssq)
                Fev(:,mu) = Fev(:,mu) + Fpair12
                Fev(:,nu) = Fev(:,nu) - Fpair12
            End Do
          End Do
      End If
  End Subroutine Excluded_volume_force
End Module BSpModel


Subroutine Time_Integrate_Chain(NBeads, R_Bead, spring_type, &
     tcur, tmax, Delts, &
     Hstar, Zstar, Dstar, L0s, &
     Q0s, & !FENE-Fraenkel Replacement
     seed1, Nsamples, times, samples)
  Use bspglocons
  Use bspglovars
  Use Flowvars
  Use bspintfacs, TIC_from_bspintfacs => Time_Integrate_Chain ...
     !FENE-Fraenkel Replacement
  Use bspmodel
  Use csputls

  !Use blas_interfaces unable to obtain proper interface
  !which allows matrices of arbitrary rank to be passed
  Implicit None

  Integer, Intent(in) :: NBeads
  Real (DBprec), Intent (inout), Dimension(:,:) :: R_Bead ! Initial ...
     positions of Beads !FENE-Fraenkel Replacement

  Integer, Intent (in) :: spring_type ! Spring force law type
  ! Hookean = 1, FENE = 2, ILC = 3, WLC = 4, FENE-Fraenkel = 5 ...
     !FENE-Fraenkel Replacement


  Real (DBprec), Intent (in) :: tcur,tmax,Delts  ! Integration time ...
     interval specs !FENE-Fraenkel Replacement
```

```
470   Real (DBprec),  Intent (in) :: Hstar         ! HI parameter ...
         (non-dim) !FENE-Fraenkel Replacement
      Real,  Intent (in) :: Zstar,Dstar   ! EV parameters (non-dim)
      Real (DBprec),  Intent (in) :: L0s           ! finite ext., param ...
         sqrt(b) !FENE-Fraenkel Replacement
      Real (DBprec),  Intent (in) :: Q0s           ! natural spring ...
         length !FENE-Fraenkel Replacement

475   Integer (k4b), Intent (inout) :: seed1 ! seed for rnd number

      Integer, Intent(in) :: Nsamples     ! Number of sampling points
      Real (DBprec),  Intent (in), Dimension(:) :: times    ! Sampling ...
         instances !FENE-Fraenkel Replacement
      Real (DBprec), Intent (inout), Dimension(:,:) :: samples ...
         !FENE-Fraenkel Replacement
480
      Real (DBprec) R_trvec(Ndim) !FENE-Fraenkel Replacement
      Real (DBprec) rtrial !FENE-Fraenkel Replacement

   !!!-----------------------------------------------------------|
485   !     This driver uses:
      !     Predictor-corrector        scheme for excluded volume
      !     Fully implicit scheme for the spring force
      !           based on Ottinger's and Somasi et al.'s suggestions
      !     Warner spring
490   !     Rotne-Prager-Yamakawa HI tensor with Chebyshev polynomial
      !         approx. using Fixman's approximation to get conditioner
      !     Narrow Gaussian EV potential
   !!!-----------------------------------------------------------|

495   ! external BLAS functions
      Real snrm2,sdot
      Real (DBprec) dnrm2, ddot !FENE-Fraenkel Replacement

      Real (DBprec) time, fd_err, fd_err_max, om1, om2, rs,& ...
         !FENE-Fraenkel Replacement
500     rsbyhs, hsbyrs, hsbyrs2, L_max, L_min, d_a, d_b,&
        gama_mag, lt_err, dtsby4, pcerr, delx2
      Real (DBprec) & !FENE-Fraenkel Replacement
```

```
        ! various scratch storages for position vector
        cofm(Ndim), & ! center of mass
505     R_pred(Ndim,NBeads), &
        DR_pred(Ndim,NBeads), &
        Ups_pred(Ndim,NBeads), &
        R_pred_old(Ndim,NBeads), &
        R_corr(Ndim,NBeads), &
510     Rtemp(Ndim,NBeads), &
        delta_R(NBeads,NBeads),&
        b2b_sup(Ndim,Nbeads,Nbeads), & ! bead to bead vector (super ...
           diagonal)
        deltaR_sup(Nbeads,Nbeads), & ! bead to bead dist (super ...
           diagonal)
        ! forces
515     F_ev(Ndim,NBeads), &     ! excluded volume forces
        F_spring(Ndim,NBeads), & ! spring forces
        F_tot(Ndim,Nbeads), &    ! total forces
        ! symmetric diffusion tensor, only upper diagonal elements ...
           stored
        Diffusion_sup(Ndim,NBeads,Ndim,NBeads), &
520     ! another scratch storage for diffusion tensor
        Dp_sym_up(Ndim,NBeads,Ndim,NBeads), &
        DelS(Ndim,NBeads),&
        kappa(Ndim,Ndim),&  !flow field divergence
        ! for Chebyshev polynomials
525     X_0(Ndim,NBeads), X_l_1(Ndim,NBeads), &
        X_l(Ndim,NBeads), X_lp1(Ndim,NBeads), &

        cheba(0:MAXCHEB), gama_mu(Ndim), &
        diffD(Ndim,NBeads-1,Ndim,NBeads), &
530     diffUps(Ndim,NBeads-1)

    ! some connector forces and related requiring higher precision
    Real (DBprec) ff,  &
        F_con_mu(Ndim), &    ! connector force between beads
535     F_con_mu1(Ndim)    ! connector force, save for mu-1

    Logical fd_check, ncheb_flag
```

```
!     Other definitions...
Integer i,  l, mu, nu, ncheb, ncheb_old, isample, lt_count
Real temp1, sqrt2inv, TPI, RPI, TRPI, &
     C1, C2, C3, C4, C5, C6, C7, RPI3by4 ! , r !FENE-Fraenkel ...
        Replacement !SEE BELOW
Real (DBprec) r !FENE-Fraenkel Replacement


!     Definitions for the BLAS-2 routine "dsymv" !FENE-Fraenkel ...
   Replacement
Integer Ndof, lda, ldadiff, incx, incy
Real (DBprec) alpha, beta !FENE-Fraenkel Replacement

sqrtb = L0s ! sqrtb will be needed by other modules
natlength = Q0s ! natural length Q0 star will be needed by other ...
   modules when
                ! doing the FENE-Fraenkel calculation ...
                   !FENE-Fraenkel Replacement

! for EV
zsbyds5 = Zstar/(Dstar**5.0)
pt5bydssq = 0.5/(Dstar*Dstar)
! For LJ
!ktbyepsilon = Zstar
!sigmabylk = Dstar

sqrt2inv = 1.0/(2.0**0.5)
TPI = 2.0*PI
RPI = Sqrt(PI)
TRPI = 2.0*RPI
C1 = TPI/3.0
C2 = 1.0
C3 = 0.75/RPI*0.375 !C3 = 9./32/RPI
C4 = 8.0*PI
C5 = 14.14855378
C6 = 1.21569221
C7 = 0.09375/RPI
RPI3by4 = RPI*0.75
```

```
575   incx = 1
      incy = 1
      Ndof = Ndim*NBeads  ! degrees of freedom
      lda = Ndof

580   fd_err_max = 0.0025
      pcerr = 0.0


      delta_R = 0.0
585

      Diffusion_sup = 0.0
      ! diagonal elements
      Forall (mu = 1:Nbeads, i = 1:Ndim )
590      Diffusion_sup(i,mu,i,mu) = 1.0
      End Forall


      !     Get initial estimates of L_max and L_min using
595   !     Kroeger et al's approx. of Zimm theory's preds.;
      !     the number of Chebyshev terms, and the Chebyshev coefficients
      L_max = 2*(1 + PI*Sqrt(Dble(NBeads))*Hstar)
      L_min = (1 - 1.71*Hstar)/2     ! Note Hstar < 0.58
      ncheb = Int(Sqrt(L_max/L_min)+0.5)+1
600   ncheb_flag = .False.
      d_a = 2/(L_max-L_min)
      d_b = -(L_max+L_min)/(L_max-L_min)
      Call chbyshv(ncheb, d_a, d_b, cheba)
      DelS = 0.0
605

      isample = 1
      delx2 = 0

      time = tcur
610

      !-----------------------------------------------------------------|
      !                The Time integration loop begins here ...
```

```
                    |
!-------------------------------------------------------------------|

615

Overtime: Do While (time.Le.Tmax+Delts/2)

   ! find the center of mass
620    cofm = 0.0
   Do mu = 1, NBeads
      cofm = cofm + R_bead(:,mu)
   End Do

625    cofm = cofm/NBeads

   ! shift the origin to the current center of mass
   Forall (mu = 1: NBeads)
      R_Bead(:,mu) = R_Bead(:,mu) - cofm
630    End Forall

   ! calculate the bead to bead vector and its magnitude
   ! for use in the forces
   Call b2bvector_sym_up(NBeads,R_Bead,b2b_sup)
635    Call modr_sym_up(NBeads,b2b_sup,deltaR_sup)


   ! change the nature of the spring, in the specific subroutine
   ! in Spring_force(), in module Bead_spring_model
640    Call Spring_force(spring_type,NBeads,b2b_sup,deltaR_sup,F_spring)
   Call Excluded_volume_force(NBeads,b2b_sup,deltaR_sup,F_ev)

   !write (*,*) "F_spring is", F_spring !FENE-Fraenkel Replacement
   !write (*,*) "F_ev is ", F_ev !FENE-Fraenkel Replacement
645
   ! Therefore, total force on each bead...
   F_tot = F_spring + F_ev

   !        Calculate the RPY diffusion tensor
650    If (Hstar.Gt.0) Then
```

```
      Do nu = 2,NBeads
         Do mu = 1,nu-1
            rs = deltaR_sup(mu,nu)
            hsbyrs = Hstar/rs
            rsbyhs = rs/Hstar
            hsbyrs2 = hsbyrs*hsbyrs
            If (rsbyhs.Ge.TRPI) Then
               om1 = RPI3by4*hsbyrs*(1.0+C1*hsbyrs2)
               om2 = RPI3by4*(hsbyrs/(rs*rs))*(1-TPI*hsbyrs2)
            Else
               om1 = 1-C3*rsbyhs
               om2 = C7*rsbyhs/(rs*rs)
            End If

            ! store only the upper triangular in the mu,nu index.
            Call tensor_prod_of_vec(om2,b2b_sup(:,mu,nu), &
                 Diffusion_sup(:,mu,:,nu) )

            ! additional factor for diagonal elements, in each mu,nu
            Forall (i = 1:Ndim )
               Diffusion_sup(i,mu,i,nu) = ...
                  Diffusion_sup(i,mu,i,nu) + om1
            End Forall
         End Do
      End Do
   End If ! Hstar

   ! mean square displacement of cofm
   ! since the bead positions are centered w.r.t. cofm at every
   ! instant, cofm is the incremental displacement
   if (time .gt. tcur) then
      delx2 = delx2 + Sum(cofm * cofm)
   end if


   !-------------------------------------------------------------|
   !       Take samples when required ...
                                        |
   !-------------------------------------------------------------|
```

```
      If (Nsamples.Gt.0) Then
690      ! ideally it shud b delts/2, but owing to precision errors
         ! we keep it slighly greater than 0.5
         If(Abs(time-times(isample)).Le.Delts*0.51) Then
            Call chain_props (NBeads, R_Bead, F_tot, ...
                samples(1:14,isample))

695         ! Save information on the error between the predictor
            ! and the final corrector
            samples(15,isample) = pcerr

            !! obtain time correlations
700         If (gdots .Eq. 0) Then
               Call time_correl(NBeads, R_Bead, F_tot, tcur, time, &
                    samples(16,isample))
            End If

705         ! Diffusivity
            If (time > 0.0) samples(17,isample) = delx2/2/Ndim/time


            isample = isample + 1
710      End If
      End If


715

      !-------------------------------------------------------------------|
      !    Chebyshev polynomial approximation of DelS begins here ...
                 |
      !-------------------------------------------------------------------|

720   !Generate the random vector X_0
      X_0 = 0.0
      Call ran_1(Ndof, X_0, seed1)
      X_0 = X_0 - 0.5
      X_0 = (X_0*X_0*C5 + C6)*X_0! Element-wise multiplications
```

```
725
      ! Has check for deviation from fluctuation-dissipation theorem
      ! been performed?
      fd_check = .False.

730   If (Hstar.Gt.0) Then


         FDloop: Do
            ! Update DelS vector
735         DelS = cheba(0) * X_0

            ! Shift the D matrix
            Dp_sym_up = d_a * Diffusion_sup

740         ! diagonal elements
            Forall (mu = 1:Nbeads, i = 1:Ndim )
               Dp_sym_up(i,mu,i,mu) = Dp_sym_up(i,mu,i,mu) + d_b
            End Forall

745         ! Calculate the second Chebyshev vector
            X_l_1 = X_0
            alpha = 1.D0 !FENE-Fraenkel Replacement
            beta = 0.D0 !FENE-Fraenkel Replacement

750         ! BLAS2 symmetric matrix-vector multiplication
            Call dsymv('U', Ndof, alpha,Dp_sym_up,lda,  X_l_1,incx, ...
                & !FENE-Fraenkel Replacement
                  beta,X_l,incy)

            ! Update DelS vector
755         DelS = DelS + cheba(1)*X_l

            Do l = 2,ncheb
               alpha = 2.D0 !FENE-Fraenkel Replacement
               beta = 0.D0 !FENE-Fraenkel Replacement
760            Call dsymv('U', Ndof, alpha,Dp_sym_up,lda, X_l,incx,  ...
                   & !FENE-Fraenkel Replacement
                     beta,X_lp1,incy)
```

```
            X_lp1 = X_lp1-X_l_1
            X_l_1 = X_l

            ! The l-th Chebyshev vector
            X_l = X_lp1

            ! Update DelS vector
            DelS = DelS + cheba(l)*X_l

        End Do

        ! Calculate the deviation from
        ! the fluctuation-dissipation theorem

        If (.Not.fd_check) Then
            fd_err = ddot(Ndof, DelS, 1, DelS, 1) ! BLAS-1 ...
                function !FENE-Fraenkel Replacement
            alpha = 1.D0 !FENE-Fraenkel Replacement
            beta = 0.D0 !FENE-Fraenkel Replacement

            !Use D:X_0X_0 = X_0.D.X_0
            !Get D.X_0 first, and then get the dot product of X_0 ...
                with the
            ! resulting vector. X_l is reused
            Call dsymv('U', Ndof,alpha,Diffusion_sup,lda, & ...
                !FENE-Fraenkel Replacement
                  X_0,incx, beta,X_l,incy)
            temp1 = ddot(Ndof, X_0, 1, X_l, 1) !FENE-Fraenkel ...
                Replacement
            fd_err = Abs((fd_err - temp1)/temp1)
        End If

        If ((fd_err.Le.fd_err_max).Or.fd_check) Then
            Exit FDloop
        Else
            !  If the fd_check has been performed and deviation ...
                is large
            !  recaclulate the number of Chebyshev terms required.
            !  First, get the maximum and minimum eigen values of D
```

```
            !  using Fixman's suggestion.
            fd_check = .True.
            Call maxminev_fi(Ndof, Diffusion_sup, L_max, L_min)
            ncheb_old = ncheb
            ncheb = Int(Sqrt(L_max/L_min)+0.5)+1
            If ((ncheb/ncheb_old).Gt.2) ncheb_flag = .True.
            If (ncheb.Gt.500) ncheb = 500
            d_a = 2/(L_max-L_min)
            d_b = -(L_max+L_min)/(L_max-L_min)
            Call chbyshv(ncheb, d_a, d_b, cheba)
         End If
      End Do FDloop
   Else ! Hstar == 0, the free-draining case
      DelS = X_0
   End If

   DelS = DelS * sqrt(delts)

   !write (*,*) "DelS is ", DelS !FENE-Fraenkel Replacement

   !-----------------------------------------------------------------------|
   !        The predictor step ...
                                                  |
   !-----------------------------------------------------------------------|


   If (Hstar.Gt.0) Then
      alpha = (2.5D-1)*Delts !FENE-Fraenkel Replacement
      beta = 0.D0 !FENE-Fraenkel Replacement
      ! Assigns DR_pred <- 0.25*Delts* D.F
      Call dsymv('U', Ndof,alpha,Diffusion_sup,lda, F_tot,incx, & ...
          !FENE-Fraenkel Replacement
           beta,DR_pred,incy)
   Else
      DR_pred = 0.25 * Delts * F_tot
   End If


   ! Add the K.R vector
```

```
      call get_kappa(time,kappa)
      DR_pred = DR_pred + Delts * Matmul(kappa,R_Bead)

!
      ! Eq (14)
      R_pred = R_Bead + DR_pred + sqrt2inv*DelS

      R_pred_old = R_pred

!
      !write (*,*) "Original bead position is ", R_Bead ...
         !FENE-Fraenkel Replacement

      !-----------------------------------------------------------------|
      !        The first semi-implicit corrector step ...
                              |
      !-----------------------------------------------------------------|

      ! initialise with part of Eq (18)
      Ups_pred = R_Bead + 0.5*DR_pred + sqrt2inv*DelS

      If (Zstar > 0) Then
         ! Calculate distances between beads using R_pred
         ! reuse variables
         Call b2bvector_sym_up(NBeads,R_pred,b2b_sup)
         Call modr_sym_up(NBeads,b2b_sup,deltaR_sup)
         Call Excluded_volume_force(NBeads,b2b_sup,deltaR_sup,F_ev)

         ! Calculate D.FEV using R_pred and update
         If (Hstar.Gt.0) Then
            alpha = 1.25D-1*Delts      ! The prefactor is 1/8 and ...
               not 1/4 !FENE-Fraenkel Replacement
            beta = 1.D0               ! Add to existing ...
               !FENE-Fraenkel Replacement
            Call dsymv('U', Ndof, alpha,Diffusion_sup,lda, ...
               F_ev,incx,  & !FENE-Fraenkel Replacement
                 beta,Ups_pred,incy)
         Else
            Ups_pred = Ups_pred + 0.125 * Delts * F_ev
         End If
      End If
```

```
!        Calculate the 0.5*Delts*K.R_pred vector

! Add the K.R vector
call get_kappa(time+delts,kappa)
Ups_pred = Ups_pred + 0.5 * Delts * Matmul(kappa,R_pred)

! Eq (18) is completely assembled now

! Generating the matrix obtained by using the D_nu
! operator on the D super-matrix

! (in the following comments indices i,j are omitted for clarity)
!    diffD(mu,nu) = D(mu+1,nu) - D(mu,nu)
! this is true only for nu > mu, since D's elements are ...
   computed only
! for nu >= mu, and the RHS depends on mu+1.  for nu <= mu+1, ...
   the RHS
! is rewritten in terms of the symmetric matrix D
!    diffD(mu,nu) = D(nu,mu+1) - D(nu,mu)
! so that all the elements of the RHS are the computed ones

Forall (mu=1:NBeads-1)
   diffUps(:,mu) = Ups_pred(:,mu+1) - Ups_pred(:,mu)
End Forall

Forall (mu=1:NBeads-1, nu=1:Nbeads, nu > mu)
   diffD(:,mu,:,nu) = Diffusion_sup(:,mu+1,:,nu) - ...
      Diffusion_sup(:,mu,:,nu)
End Forall
Forall (mu=1:NBeads-1, nu=1:Nbeads, nu <= mu)
   diffD(:,mu,:,nu) = Diffusion_sup(:,nu,:,mu+1) - ...
      Diffusion_sup(:,nu,:,mu)
End Forall


! Start the loop for solving for connector vectors
R_corr = R_pred
```

```
      ldadiff = Ndim*(Nbeads-1)

      lt_count = 0

      dtsby4 = Delts/4.0D0 !FENE-Fraenkel Replacement


      Keepdoing: Do

         F_con_mu1 = 0.0

         oversprings: Do mu = 1,NBeads-1


            ! the connector force for this spring is obtained from
            ! Fs(mu) = Fc(mu) - Fc(mu-1)

            F_con_mu = F_con_mu1 + F_spring(:,mu)
            ! note:  F_spring contains forces evaluated with the
            !          predictor Q for all beads < mu
            !          corrector Q for all beads > mu

            !write (*,*) "F_con_mu is ", F_con_mu !FENE-Fraenkel ...
               Replacement

            If ( Hstar.Gt.0 ) Then
               gama_mu = 0.125*Delts* &
                    Matmul( &
                    Reshape(diffD(:,mu,:,:),(/ Ndim, Ndim*Nbeads /) ...
                       ),  &
                    Reshape(F_spring,             (/ Ndim*Nbeads /)) &
                    )
            Else
               gama_mu  = 0.125*Delts*(F_spring(:,mu+1) - ...
                  F_spring(:,mu))
            End If

            !write (*,*) "Spring force on first bead is ", ...
               F_spring(:,mu) !FENE-Fraenkel Replacement
```

```
!write (*,*) "Spring force on second bead is ", ...
   F_spring(:,mu+1) !FENE-Fraenkel Replacement

! the remaining terms on the RHS of Eq.(20)
gama_mu = gama_mu + diffUps(:,mu) + 0.25 * F_con_mu * Delts

gama_mag = dnrm2(Ndim,gama_mu,1) !FENE-Fraenkel ...
   Replacement ! BLAS DOUBLE normal two

!FENE-Fraenkel Replacement BELOW
R_trvec = R_bead(:,mu+1) - R_bead(:,mu)
!write (*,*) "R_trvec is ", R_trvec
rtrial = (dnrm2(Ndim,R_trvec,1))/sqrtb
!write (*,*) "Original value of r (trial) is ", rtrial
!FENE-Fraenkel Replacement ABOVE

! r = Q_nu_mag/sqrt(b) varies from (0,1)
Call solve_implicit_r(spring_type,dtsby4,gama_mag/sqrtb, &
     natlength/sqrtb,r,ff,rtrial) !FENE-Fraenkel Replacement
!Write(*,*) "Output of solve_imp_r is ff = ", ff ...
   !FENE-Fraenkel Replacement
!write(*,*) 'Press Enter to continue' !FENE-Fraenkel ...
   Replacement
!read(*,*) !FENE-Fraenkel Replacement
! implicit solution of Eq.(21)
!   r ( 1 + deltat/4 ff) - |gama_mu|/sqrtb == 0
! and returns r and ff, the factor in the spring force other
! than hookean F = H Q ff

if (spring_type .eq. FENEFraenkel) then !FENE-Fraenkel ...
   case !FENE-Fraenkel Replacement
   if ((r - (natlength/sqrtb)) .gt. (1. - (1e-6))) then
      r = (natlength/sqrtb) + 1. - 1e-6
   end if
   if ((r - (natlength/sqrtb)) .lt. ((1e-6) - 1.)) then
      r = (natlength/sqrtb) - 1. + 1e-6
   end if
else !All other cases (no natural length) !FENE-Fraenkel ...
   Replacement
```

```
970               if (Abs(r-1.) .Lt. 1e-6) r = 1 - 1e-6
                  end if
                  !If value for r gets too close to limits of the range of ...
                     allowable lengths,
                  !between (natlength/sqrtb) - 1 and (natlength/sqrtb) + 1,
                  !then force it within range.
975
                  ! unit vector for Q_mu same as for Gama_mu
                  gama_mu = gama_mu/gama_mag
                  gama_mu = gama_mu*r*sqrtb; ! connector vector update

980               ! corrector positions vector update
                  R_corr(:,mu+1) = R_corr(:,mu) + gama_mu

                  ! Connector force
                  F_con_mu1 = gama_mu * ff ! to b used for next spring
985
                  ! update the spring forces, and connector force for this ...
                     spring
                  ! which will b used at the start of the loop
                  F_spring(:,mu)   = F_spring(:,mu)   + (F_con_mu1 - F_con_mu)
                  F_spring(:,mu+1) = F_spring(:,mu+1) - (F_con_mu1 - F_con_mu)
990
            End Do oversprings

            Rtemp = R_corr - R_pred
            !lt_err = snrm2(Ndof,Rtemp,1)/NBeads
995         lt_err = dnrm2(Ndof,Rtemp,1)/dnrm2(Ndof,R_Bead,1) ...
                !FENE-Fraenkel Replacement !Previously snrm2

            lt_count = lt_count + 1

            !If ...
                ((lt_err.Lt.imploop_tol).Or.(lt_count.Gt.2*Nbeads*Nbeads)) ...
                Exit
1000        !If ((lt_err.Lt.imploop_tol)) Exit
            If ((lt_err.Lt.imploop_tol).Or.(lt_count > 10*Nbeads)) Exit

            R_pred = R_corr
```

```
      End Do Keepdoing

      !write (*,*) "Updated bead position is ", R_corr ...
         !FENE-Fraenkel Replacement

!     if (lt_count > 10*NBeads) write (*,1024) lt_count, Delts , ...
   Gdots*time !EXTRA OUTPUT
!1024 format ('Loop exceeded ', I3, ' for dt = ', F11.4, ' at ...
   strain ', G11.4) !EXTRA OUTPUT
!write (*,1024) lt_count, Delts , Gdots*time
      ! Final Major updates, new position vector and time
      R_Bead = R_corr
      time = time + Delts

      Rtemp = R_Bead - R_pred_old
      pcerr = snrm2(Ndof,Rtemp,1)/NBeads
      pcerr = snrm2(Ndof,Rtemp,1)/snrm2(Ndof,R_Bead,1)

      If (ncheb_flag) Then
         ncheb = ncheb_old
         ncheb_flag = .False.
      End If
  End Do    Overtime


End Subroutine Time_Integrate_Chain
```

D.5 properties.f90

```
!!! Time-stamp: <properties.f90 15:32, 03 May 2018 by DP Amarasinghe>

!-----------------------------------------------------------------
!   utilities to extract properties from a given configuration
!-----------------------------------------------------------------

!!! $Log: properties.f90,v $
!!! Revision 1.1  2004/01/29 22:12:18  pxs565
!!! Initial revision
!!!



Subroutine chain_props(NBeads, Rbead, F, props)
  use bspglocons
  use bspglovars
  Implicit None
  Integer, intent (in) :: NBeads
  Real (DBprec), intent (in), dimension(:,:)  :: Rbead, F ...
     !FENE-Fraenkel Replacement
  Real (DBprec), intent (out),dimension(:) ::props !FENE-Fraenkel ...
     Replacement

  !-----------------------------------------------------------------------
  !    This routine estimates the contributions of a single bead-spring
  !    chain in the configuration specified by the vector R, to the
  !    equilibrium and non-equilibrium properties of the solution.
  !    (Time correlations are evaluated by a separte procedure.)
  !
  !    The idea behind using this routine is to have a code wherein
  !    the main program can be run using different property estimation
  !    routines (such as this) without any serious changes to the
  !    main program itself. In addition, the user should have the
  !    freedom to choose properties of interest.
  !
```

```
!    In the present routine, the properties are returned in the
!    vector props whose elements are given below:
!
!    props(1)  ---  square of end-to-end vector
!    props(2)  ---  1,1 component of shape tensor
!    props(3)  ---  1,2 component of shape tensor
!    props(4)  ---  1,3 component of shape tensor
!    props(5)  ---  2,2 component of shape tensor
!    props(6)  ---  2,3 component of shape tensor
!    props(7)  ---  3,3 component of shape tensor
!    props(8)  ---  First normal stress difference
!    props(9)  ---  Second normal stress difference
!    props(10) ---  1,2 component of stress tensor
!    props(11) ---  Stretch in 1 direction
!    props(12) ---  stretch in 2 direction
!    props(13) ---  stretch in 3 direction
!    props(14) ---  radius of gyration
!-----------------------------------------------------------------------
Integer i, j, k, mu, nu
Real R(Ndim,NBeads), Rc(Ndim),dist(Ndim)



props = 0.0
R = RBead

!     Calculation of square of end-to-end vector

dist = R(:,Nbeads) - R(:,1)

props(1) = sum(dist*dist)


!     Calculation of shape tensor

!       First, calculate position of centre of mass
Rc = 0.0
```

```
Do mu = 1, NBeads
   Rc = Rc + R(:,mu)
End Do
Rc = Rc/NBeads

! shift the origin to the current center of mass
Forall (mu = 1: NBeads)
   R(:,mu) = R(:,mu) - Rc
End Forall



!       Calculate shape tensor
! props (2:7)
Do nu = 1, NBeads
   k = 2
   Do i = 1,Ndim              ! Since the tensor is symmetric
      Do j = i,Ndim           ! calculate only the upper diagonal ...
          components
         props(k) = props(k) + R(i,nu) * R(j,nu)
         k = k + 1
      End Do
   End Do
End Do
props(2:7) = props(2:7)/NBeads



!     Calculation of polymer's contribution to the stress tensor
!     using the Kramers-Kirkwood expression - Eq. C in Table ...
   15.2-1 of DPL - II

! props (8:10)
Do nu = 1, NBeads
   k = 8
   props(k) = props(k) + R(1,nu) * F(1,nu) - R(2,nu) * F(2,nu)
   k = k + 1
   props(k) = props(k) + R(2,nu) * F(2,nu) - R(3,nu) * F(3,nu)
   k = k + 1
   props(k) = props(k) + R(1,nu) * F(2,nu)
End Do
```

```
110
     ! props (11:13)
     !     Calculation of "stretch" in x, y and z directions
     Do i = 1,Ndim
        props(10+i) = Maxval(R(i,:)) - Minval(R(i,:))
115  End Do

     ! prop (14)
     ! radius of gyration
     props(14) = 0.0
120  Do mu = 1, NBeads
        props(14) = props(14) + sum(R(:,mu)*R(:,mu))
     end Do
     props(14) =  props(14)/NBeads

125


   !!$        write (*,98) R,F
   !!$ 98     format (3(F10.6,1x))
   !!$        write (*,72) props(1:10)
130 !!$72      format (5(E10.4,2x))

     !     ENSURE THAT THE NProps PARAMETER IN MODULE globals IS ...
        GREATER THAN
     !     THE NUMBER OF PROPERTIES DEFINED

135 End Subroutine chain_props



   Subroutine time_correl(NBeads, R, F, t0, t, correl)
140  use bspglocons
     use bspglovars
     Implicit None
     Integer, intent (in) :: NBeads
     Real (DBprec), intent (in), dimension(:,:)  :: R, F ...
        !FENE-Fraenkel Replacement
145  Real (DBprec), intent (out)  :: correl !FENE-Fraenkel Replacement
     Real (DBprec), intent (in) :: t0, t !FENE-Fraenkel Replacement
```

```
!Real R(3*NBeads), F(3*NBeads), correl(:), t0, t

!-----------------------------------------------------------------------c
!    This routine estimates time correlations between properties ...
       c
!    at times t0 and t. ...
                                                  c
! ...
                                                                      ...
   c
!    In the present routine, the correlations are returned in the ...
      c
!    vector correl whose elements are given below: ...
                    c
! ...
                                                                      ...
   c
!    correl     ---  the autocorrelation for calculation of ...
           c
!                    linear viscoelastic properties in simple ...
   shear  c
!-----------------------------------------------------------------------c
Integer  nu, dir
Real Rc(Ndim), temp
Real, Save :: t0props

correl = 0.0

!     First, calculate position of centre of mass
Rc = 0.0
Do nu = 1, NBeads
   Rc = Rc + R(:,nu)
End Do
Rc = Rc/NBeads

!     Calculation of Sxy
temp = 0.0
dir = 1
Do nu = 1,NBeads
```

```
      temp = temp + (R(dir,nu)-Rc(dir)) * F(dir+1,nu)
    End Do

    If (t.Eq.t0) t0props = temp

180

    correl = temp*t0props


    !     ENSURE THAT THE NCorrels PARAMETER IN MODULE globals IS ...
       GREATER THAN
    !     OR EQUALT TO THE NUMBER OF PROPERTIES DEFINED
185

  End Subroutine time_correl
```

Appendix E

Vesicle preparation and method development

As mentioned in Section 5.5, the vesicle preparation protocol was examined and assessed to see if changes needed to be made to improve the quality of vesicles produced and therefore enhancing the LD signal from DPH molecules within the vesicle membranes.

The use of sodium phosphate buffers instead of water, as desrbied above, also had an effect in preventing the formation of vesicles with diameters smaller than 50 nm. Other variables were also considered including the method of manufacture of the vesicles, the effects of pumping the solution through the microfludic devices, and the use of pure lipids instead of soy bean PC. These are discussed below.

E.1 Vesicle size during manufacture

The effect of individual stages of the vesicle production process were investigated to see if any particular step was responsible for the creation of vesicles too small to be affected by fluid flows. The original vesicle production protocol was modified in the following ways:

- The use of 10 mM pH7.4 sodium phosphate buffer instead of water as the solvent to redissolve the lipid film lining the round-bottomed flask after the rotary evaporation stage.

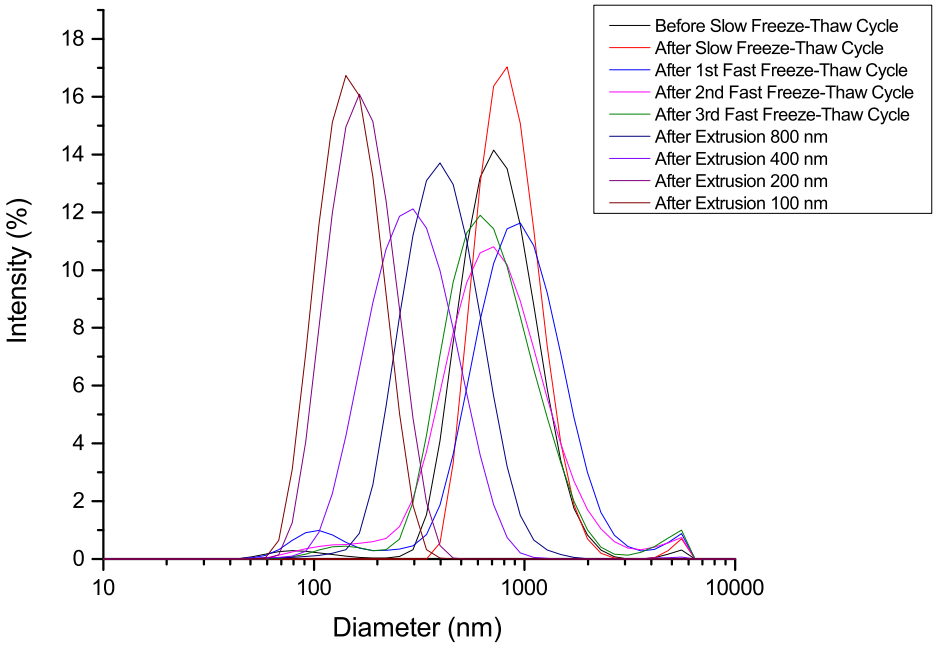


Figure E.1: Vesicle diameter distribution data (measured using intensity measurements obtained with DLS) for soy bean PC vesicles after individual stages of the manufacture protocol.

- During the preliminary experiments, it was noted that the forces required to push the vesicle solution through the extruder were very high. To overcome this, the vesicle solution concentration was reduced from 20 mg ml^{-1} to 10 mg ml^{-1}.
- Vesicle solutions were still subjected to one slow and three fast freeze-thaw cycles. The extrusion process was extended to four separate succesive extrusion subprocesses. Each subprocess consisted of 11 passes of the solution through a membrane with a particular pore-diameter size: 800 nm, 400 nm, 200 nm and 100 nm.

Table E.1: Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 25 samples at each stage.

Protocol Stage	Mean / nm	Standard Deviation / nm
Before freezing	826	81
After slow freeze-thaw	930	140
After 1st fast freeze-thaw	1055	97
After 2nd fast freeze-thaw	890	140
After 3rd fast freeze-thaw	850	150
After extrusion through 800 nm pores	440	37
After extrusion through 400 nm pores	330	23
After extrusion through 200 nm pores	177	3.1
After extrusion through 100 nm pores	155	3.1

Figure E.1 shows the measured vesicle diameters after each stage of the protocol, using the intesities measured by DLS. Table E.1 shows the mean vesicle diameters averaged across 25 sample measurments along with the respective standard deviations.

The freeze-thaw cycles maintained the large vesicle sizes, but the large standard deviations in Table E.1 and the spread of the size distributions shown in Figure E.1 indicate a wide range of vesicle sizes present, from small vesicles with diameters under 100 nm to large vesicles with diameters over 5000 nm.

With extrusion, as expected, the larger vesicles were broken down, but two issues were noted. First, the mean vesicle diameter after extrusion through a membrane with 100 nm diameter pores was greater than 100 nm. This indicated that a leak was likely to have occured during the process, probably due to a small tear in the membrane as a result of the pressures involved. Second, while extrusion through membranes with 100 nm and 200 nm pores did result in narrower vesicle size distributions than those seen for extrusion through membranes with larger pores, this also resulted in the production of vesicles as low as 60 nm. This suggested that extrusion using a membrane with 400 nm pores would be more practical, because it would produce vesicles mostly between 200 nm and 300 nm in diameter — small enough that they would be practical for use in CD experiments if membrane proteins were used; but also large enough to allow fluid flows to deform and orient vesicles more easily, thus enabling an LD signal from molecules embedded in their membranes to be more easily detected. Further investigations were carried out to establish the effect of freeze-thaw cycles on vesicle sizes. While the main reason for perfoming these cycles is to create unilamellar vesicles, it was thought to see if the number of cycles (fast or slow) should be reduced.

E.1.1 Slow cycles

A solution of 10 mg ml^{-1} soy bean PC vesicles in 10 mM pH7.4 sodium phosphate buffer was prepared and underwent cycles of slow-freezing in the freezer at −19 °C followed by thawing to room temperature without use of a heater.

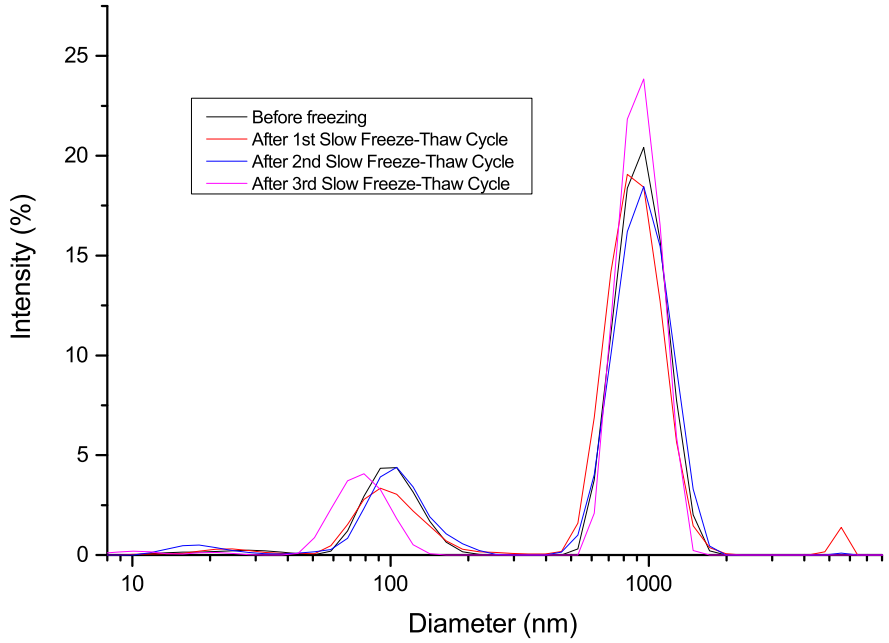


Figure E.2: Vesicle diameter distribution data (measured using intensity measurements obtained with DLS) for soy bean PC vesicles after individual stages of the manufacture protocol.

Table E.2: Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 14 samples after each slow freeze-thaw cycle.

Protocol Stage	Mean / nm	Standard Deviation / nm
Before freezing	780	91
After 1st slow freeze-thaw	830	160
After 2nd slow freeze-thaw	790	150
After 3rd slow freeze-thaw	780	58

While Table E.2 indicates that the overall mean vesicle diameter was decreasing with each succesive cycle, all slow freeze-thaw cycles, as shown in Figure E.2 resulted in a wide range of vesicle sizes. The vesicle diameters measured became more consistent with each succesive freeze-thaw cycle, as indicated by the reduction in the standard deviation. However, since the process was shown to maintain the wide range of vesicle sizes with each cycle, and performing succesive slow cycles was deemed impractical, it was decided that the protocol should still retain one slow freeze-thaw cycle.

E.1.2 Fast cycles

A solution of 10 mg ml^{-1} soy bean PC vesicles in 10 mM pH7.4 sodium phosphate buffer was prepared. This was first frozen in a freezer at −19 °C followed by thawing to room temperature without use of a heater. Cycles of fast-freezing in a bath of ethanol and dry ice followed by thawing to room temperature without heating were performed on samples of the vesicle solution.

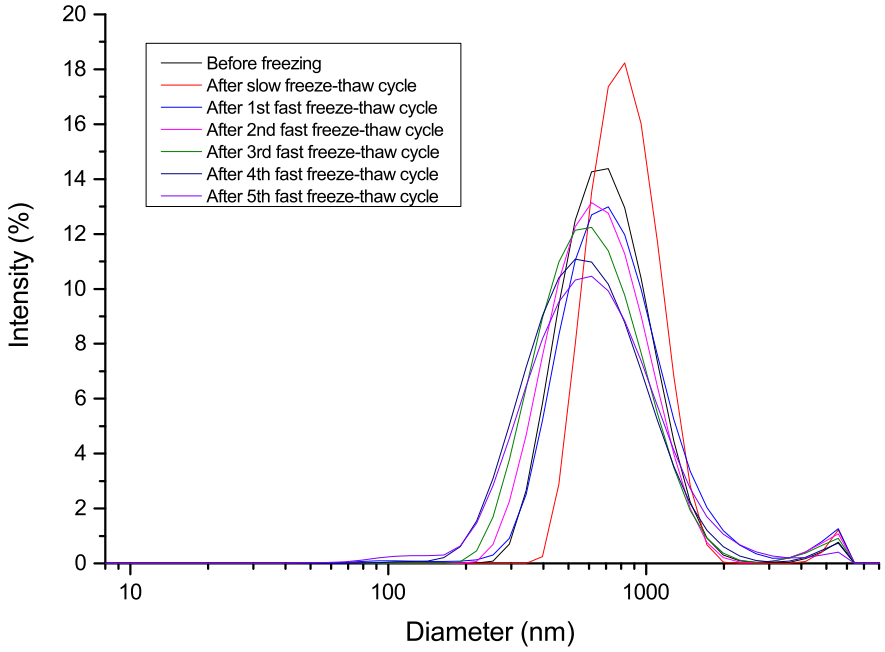


Figure E.3: Vesicle diameter distribution data (measured using intensity measurements obtained with DLS) for soy bean PC vesicles after individual stages of the manufacture protocol.

Table E.3: Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 15 samples after each slow freeze-thaw cycle.

Protocol Stage	Mean / nm	Standard Deviation / nm
Before freezing	810	58
After slow freeze-thaw	930	71
After 1st fast freeze-thaw	920	130
After 2nd fast freeze-thaw	810	60
After 3rd fast freeze-thaw	770	65
After 4th fast freeze-thaw	730	33
After 5th fast freeze-thaw	754	72

As for the slow freeze-thaw cycles, Table E.3 indicates that the overall mean vesicle diameter was decreasing with each succesive cycle, with the exception of the last cycle.

However, Figure E.3 indicates that the spread of vesicle diameters increased with successive cycles. This would suggest that a large number of fast-freeze thaw cycles should be avoided, so as to maintain a narrow distribution of vesicle sizes before and after extrusion. Furthermore, the standard deviations of mean diameter measurements were lower than those recorded for the slow cycles, indicating that the fast cycles were more reliable. It was therefore decided that the number of fast cycles in the protocol should be kept at three.

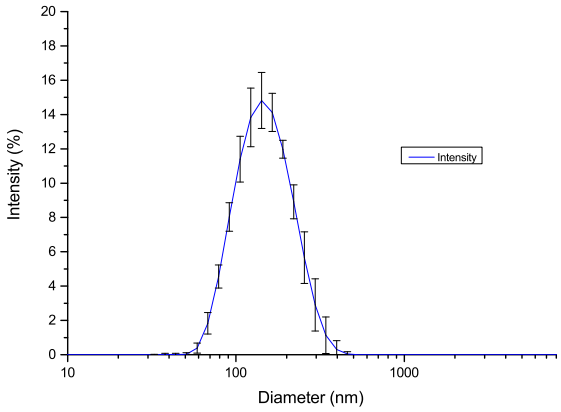
E.2 Using purified lipids

While the use of purified lipids was not taken forward for main experiments, in light of the problems observed with soy bean PC vesicles in the microfluidic experiments, further tests were perfomed to see if purified lipids would be better suited to work with DPH and the vesicle extrusion process. Two lipids were investigated — DOPC and POPC.

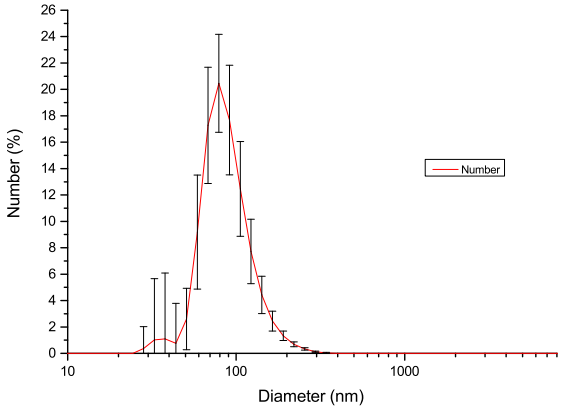
E.2.1 Using DOPC

A solution of 20 mg ml^{-1} DOPC vesicles in 10 mM pH7.4 sodium phosphate buffer with 1% DPH by mass was prepared using the protocol of one slow freeze-thaw cycle, three fast freeze-thaw cycles and extrusion through a membrane with 100 nm diameter pores.

The vesicle size distribution plots in Figure E.4 show that diameters ranged between 20 nm and 400 nm, with a mean diameter of 156.34 nm. However the absorbance spectra of two samples in Figure E.5 indicate that DPH was not taken up by the DOPC vesicles in significant amounts to show discernable peaks at 339 nm, 355 nm, 373 nm, 270 nm and 230 nm corresponding to the molecule. This could potentially be due to the low transition temperature of the lipid (−17 °C). At room temperature, the vesicle membranes would be more flexible as a result, leading to DPH molecules being less likely to be retained in the membrane.



(a) Size by intensity for DOPC vesicles.



(b) Size by number for DOPC vesicles.

Figure E.4: Vesicle size distribution data obtained using DLS for DOPC vesicles with 1.0% DPH by mass after one slow and three fast freeze-thaw cycles and extrusion through membrane with 100 nm diameter pores. Error bars indicate the standard deviation across 21 samples.

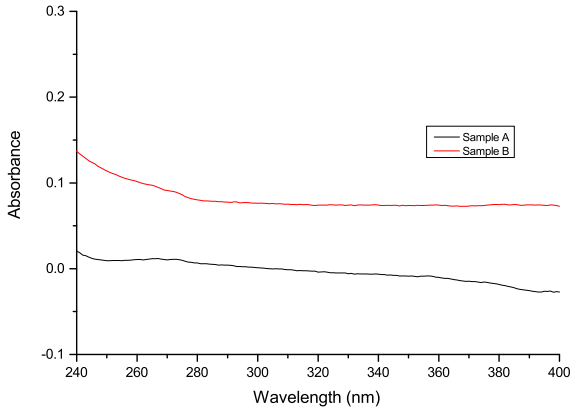


Figure E.5: Absorbance spectra two samples of DOPC vesicles with 1% DPH by mass in Couette shear flow.

E.2.2 Using POPC

A solution of 10 mg ml^{-1} DOPC vesicles in 10 mM pH7.4 sodium phosphate buffer with 1% DPH by mass was prepared using the protocol of one slow freeze-thaw cycle, three fast freeze-thaw cycles and extrusion through a membrane with 100 nm diameter pores.

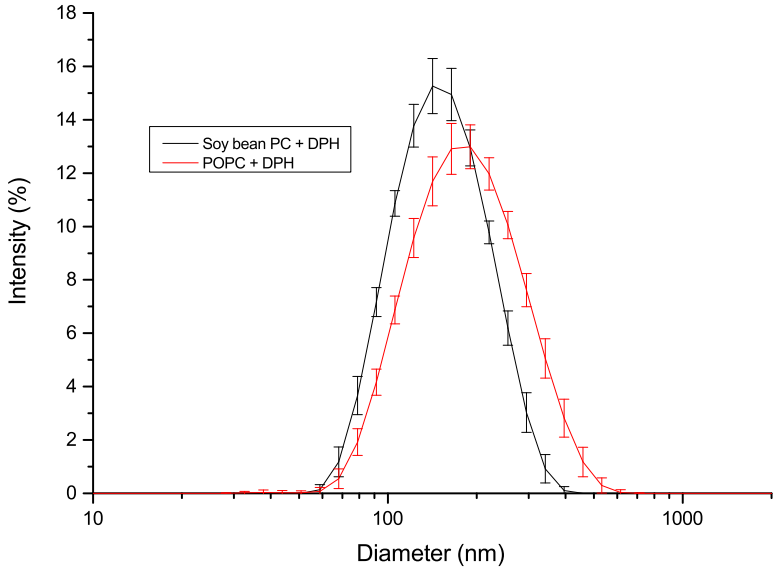
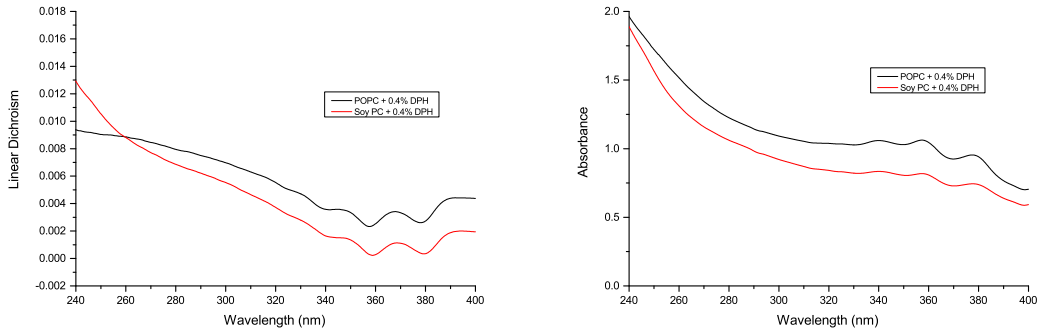


Figure E.6: Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles and POPC vesicles with 0.4% DPH by mass. Error bars indicate the standard deviation across 25 samples.

Table E.4: Mean vesicle diameters and associated standard deviations, based upon DLS intensity measurements of 25 samples after extrusion.

Lipid vesicle type	Mean / nm	Standard Deviation / nm
Soy bean PC & DPH	159	2.8
POPC & DPH	200	19

Figure E.6 indicate that the range of vesicle diameters obtained using POPC is slightly wider than that obtained with soy bean PC, ranging between 50 nm and 700 nm. The average POPC vesicle diameter is also larger than that obtained with soy bean PC, as Table E.4 shows. However, the reliability of the vesicles produced with soy bean PC appeared to be better, with a lower standard deviation in the average vesicle diameter measured.



(a) Linear Dichroism **(b)** Absorbance

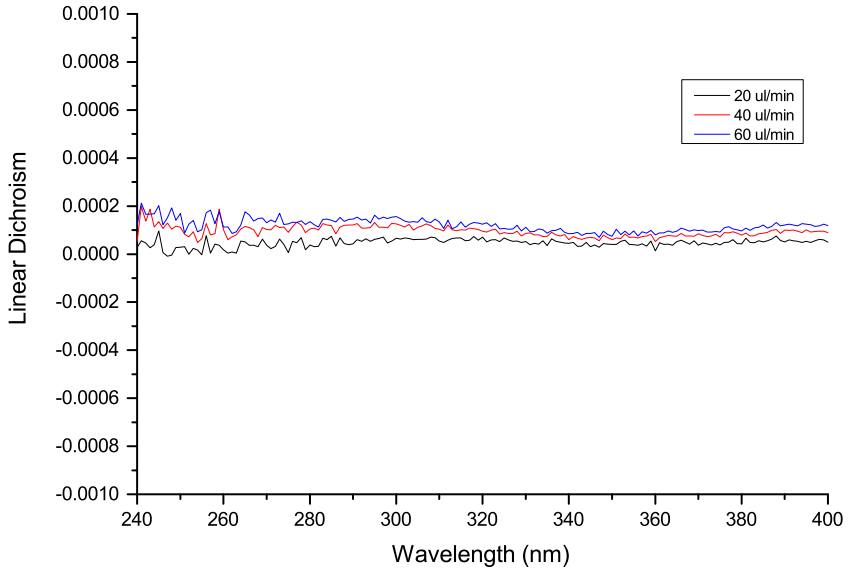
Figure E.7: Linear dichroism and absorbance spectra from Couette flow for soy bean PC and POPC vesicles with 0.4% DPH by mass, prepared after one slow and three fast freeze-thaw cycles and extrusion through membrane with 100 nm diameter pores.

Studying the spectroscopic data in Figure E.7, vesicles of both lipid types appear to retain DPH in the membrane and, in Couette flow, deform to allow an LD signal from embedded DPH molecules to be measured. While the baselines of the two lipid types are not the same, it is possible to see that the strength of the negative LD signals from DPH are similar in both vesicle types, even though the strength of the absorbance signal is greater in POPC. This could be attributed to the higher transition temperature of soy bean PC (roughly 25 °C) over POPC (−2 °C). The higher transition temperature means that the vesicle membranes are more likely to retain DPH molecules in the membrane, due to the lower flexibility in the gel state. Furthermore, as the membrane is more rigid, the DPH molecules are more likely to be oriented as the vesicle membrane does not flex easily, therefore leading to a stronger LD signal.

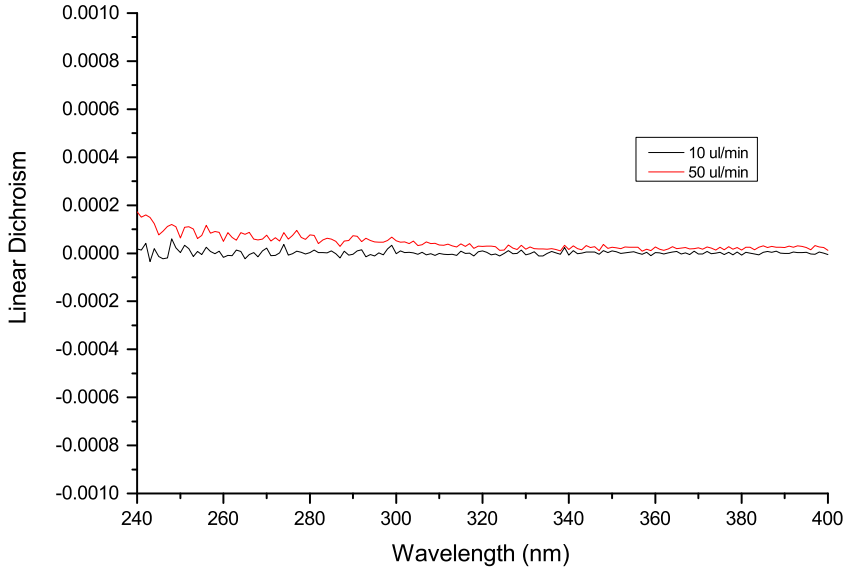
Appendix F

Supplementary figures from experiments with vesicles in Chapter 5

F.1 Additional figures referenced in Section 5.3

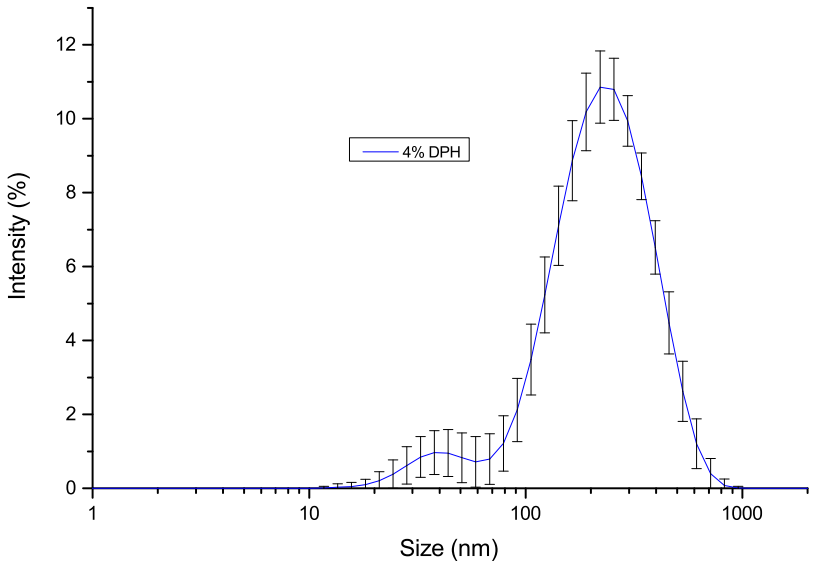


(a) Vesicles with 4% DPH by mass

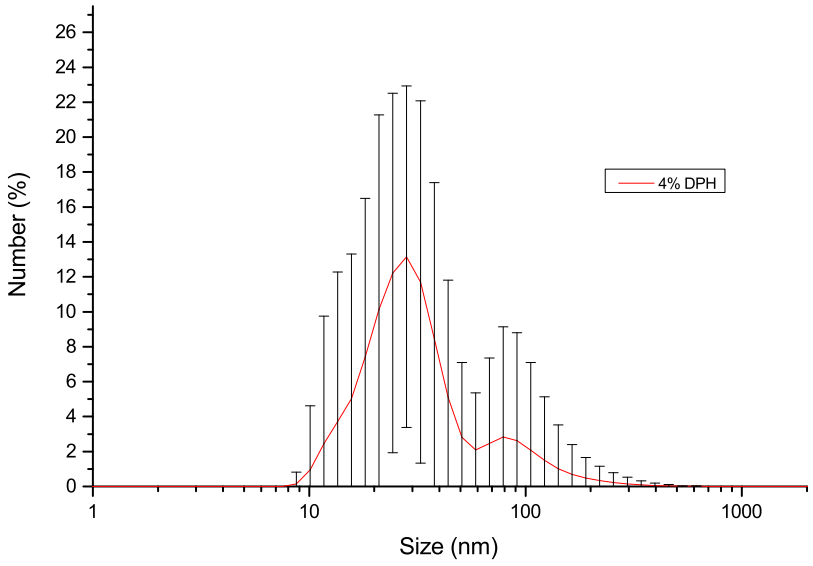


(b) Vesicles with 10% DPH by mass

Figure F.1: LD spectra of soy bean PC vesicles containing DPH (4% and 10% by mass) in pressure-driven flow in the wide channel microfluidic device.

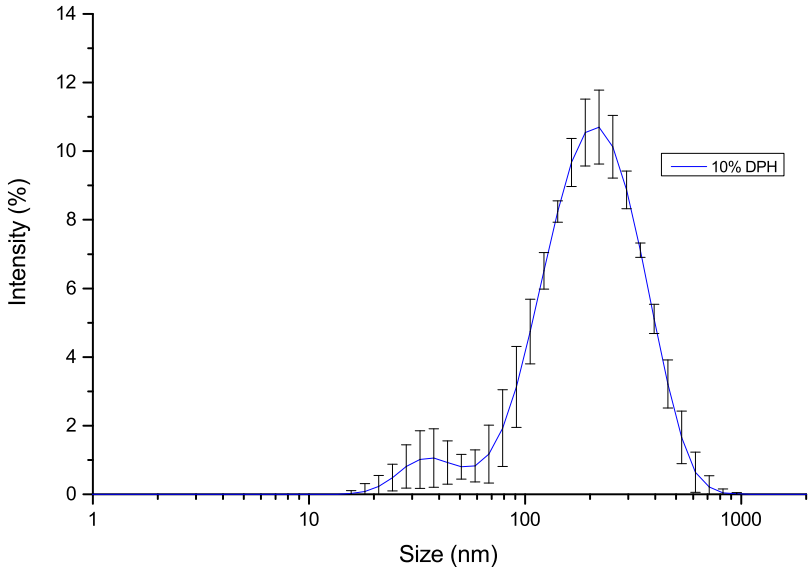


(a) Size by intensity for vesicles with 4% DPH by mass.

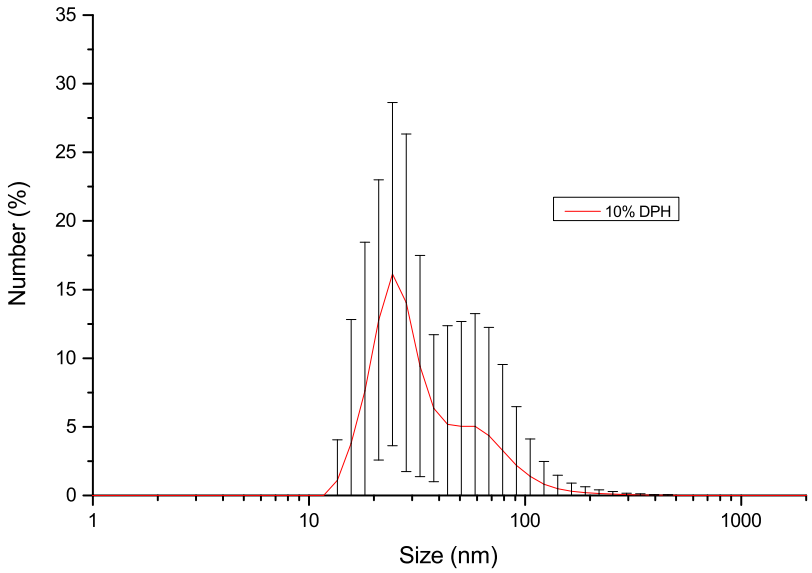


(b) Size by number for vesicles with 4% DPH by mass.

Figure F.2: Vesicle size distribution data obtained using DLS for soy bean PC vesicles containing 4% DPH by mass before use in flow experiments.



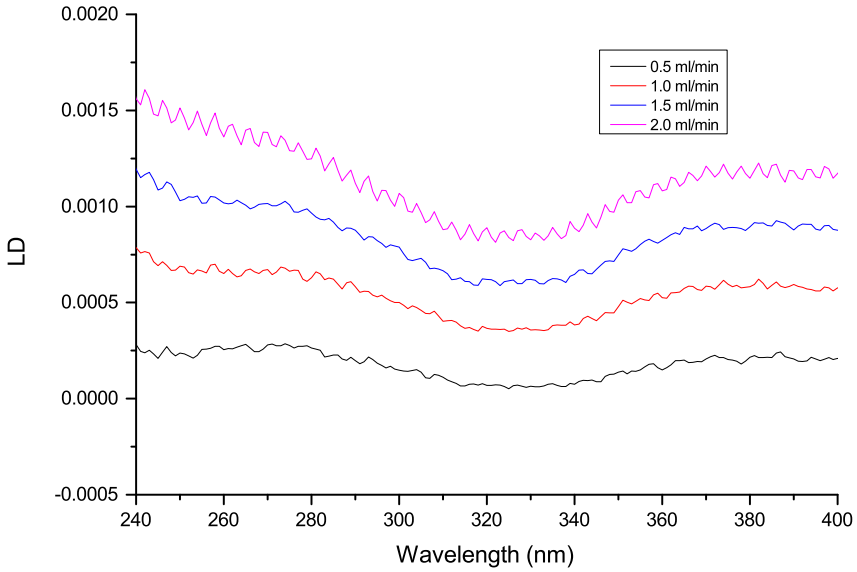
(a) Size by intensity for vesicles with 10% DPH by mass.



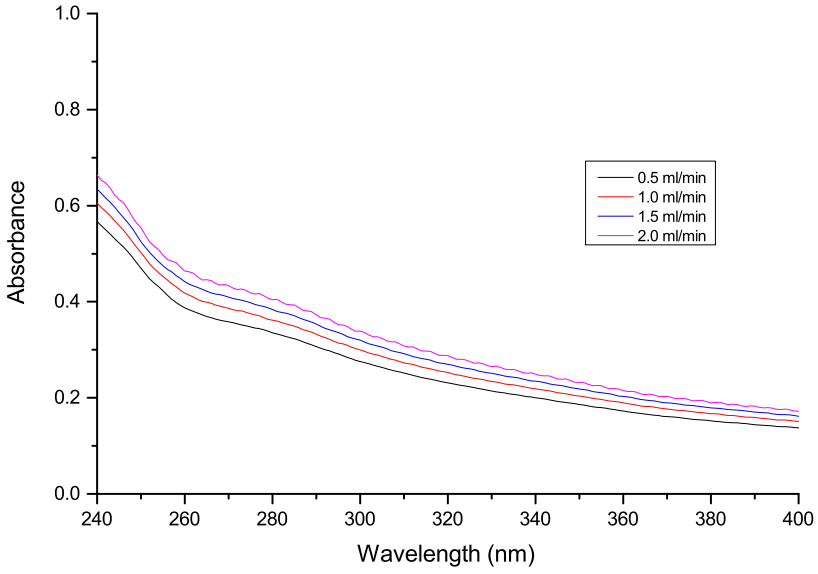
(b) Size by number for vesicles with 10% DPH by mass.

Figure F.3: Vesicle size distribution data obtained using DLS for soy bean PC vesicles containing 10% DPH by mass before use in flow experiments.

F.2 Data for vesicles without DPH in pressure-driven flow for Subsection 5.6.2



(a) Linear dichroism



(b) Absorbance

Figure F.4: Linear dichroism and absorbance spectra of soy bean PC vesicles in pressure-driven flow through the deep wide channel microfluidic device at different flow rates.

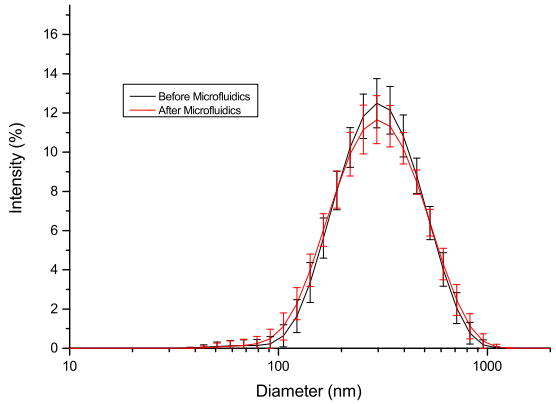
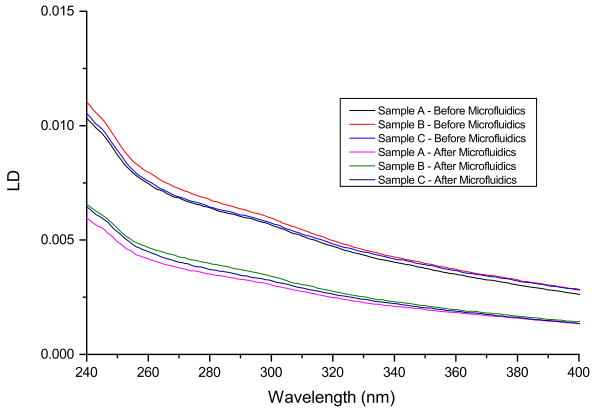
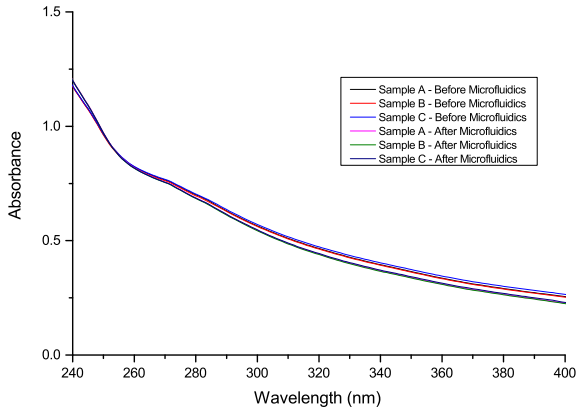


Figure F.5: Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles, before and after use in the deep wide channel microfluidic device.



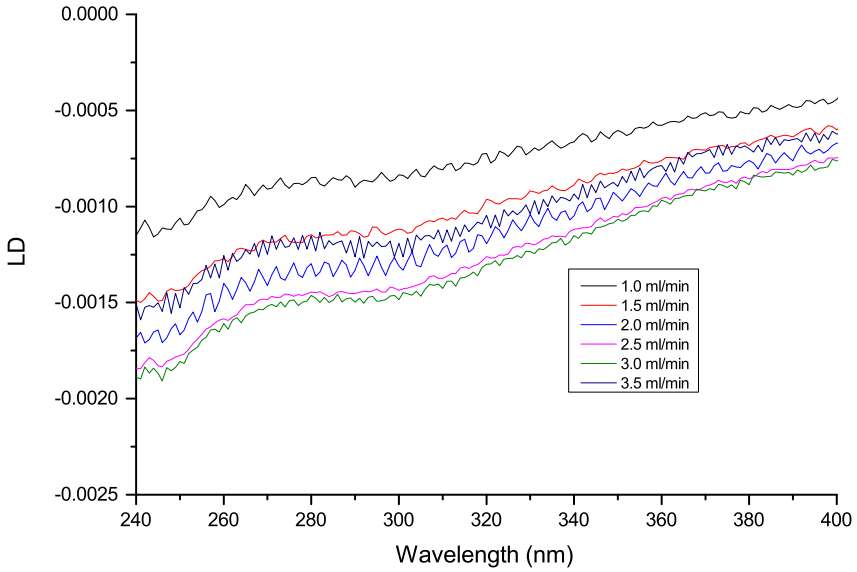
(a) Linear dichroism



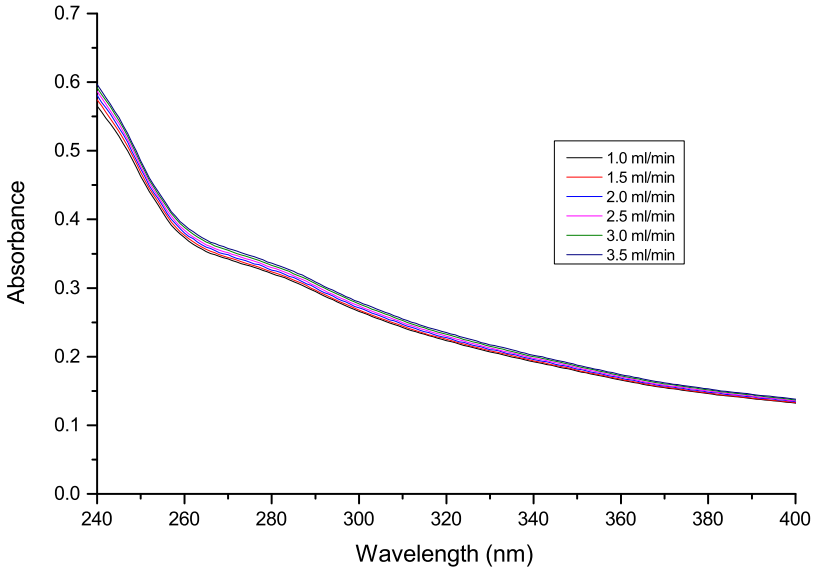
(b) Absorbance

Figure F.6: Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles, before and after use in the deep wide channel microfluidic device.

F.3 Data for vesicles without DPH in extensional flow for Subsection 5.6.3



(a) Linear dichroism



(b) Absorbance

Figure F.7: Linear dichroism and absorbance spectra of soy bean PC vesicles in extensional flow through the deep cross-slot microfluidic device at different flow rates.

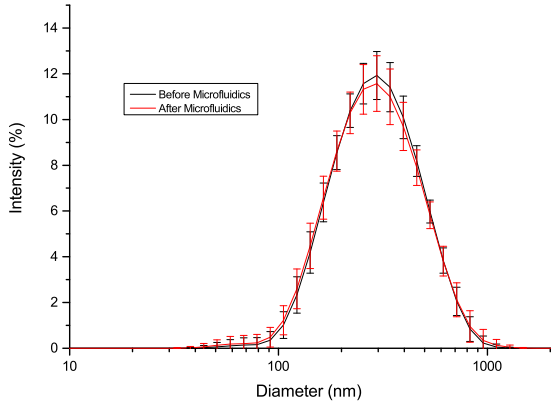
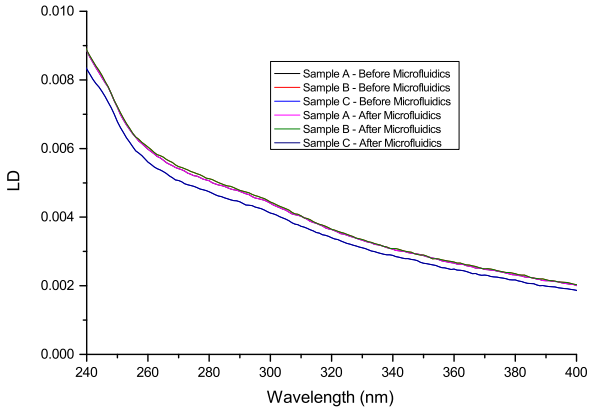
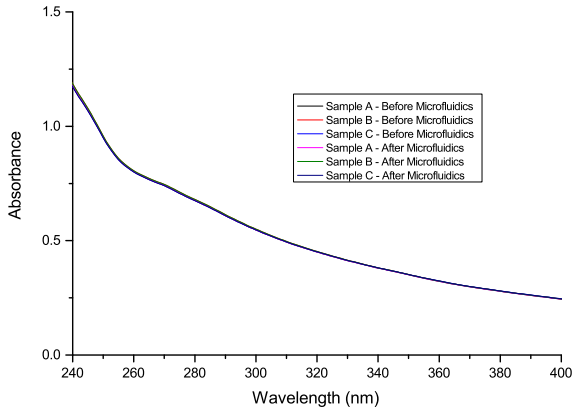


Figure F.8: Vesicle size distribution data (based on intensity measurements) obtained using DLS for soy bean PC vesicles, before and after use in the deep cross-slot microfluidic device.



(a) Linear dichroism



(b) Absorbance

Figure F.9: Linear dichroism and absorbance spectra from Couette flow for soy bean PC vesicles, before and after use in the deep cross-slot microfluidic device.